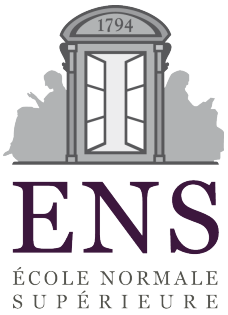
Conception et optimisation de mécanismes cryptographique anonymes

Olivier Sanders



DI/ENS - École Doctorale de Sciences Mathématiques de Paris Centre

Conception et Optimisation de Mécanismes Cryptographiques Anonymes

Thèse

présentée et soutenue publiquement le 24/09/2015

pour l'obtention du

Doctorat de l'École Normale Supérieure
(spécialité Informatique)

par

Olivier Sanders

devant le jury composé de :

Directeur de thèse :	David Pointcheval	(École Normale Supérieure)
Co-Encadrant :	Sébastien Canard	(Orange Labs)
Rapporteurs :	Jean-Sébastien Coron	(Université du Luxembourg)
	Benoît Libert	(École Normale Supérieure de Lyon)
Examinateurs :	Michel Abdalla	(École Normale Supérieure)
	Pierre-Alain Fouque	(Université de Rennes 1)
	Fabien Laguillaumie	(École Normale Supérieure de Lyon)
	Gilles Zémor	(Université Bordeaux 1)

Travaux effectués au sein de l'Applied Crypto Group des Orange Labs
et de l'Équipe de Cryptographie de l'École Normale Supérieure

Remerciements

Avant de commencer ce mémoire, je souhaite remercier tous ceux qui ont contribué, de près ou de loin, à ma thèse. Je débuterai par exprimer toute ma gratitude aux deux rapporteurs de cette thèse, Jean-Sébastien Coron et Benoît Libert, qui ont accepté de consacrer une partie de leur été à la relecture de ce mémoire. Je suis également très reconnaissant envers Michel Abdalla, Pierre-Alain Fouque, Fabien Laguillaumie et Gilles Zémor pour avoir accepté d'être membre de mon jury de thèse.

Je tiens à remercier spécialement Gilles Zémor pour m'avoir accepté au sein du Master CSI de Bordeaux et pour avoir facilité mon inscription en cours d'année en me délivrant un « sauf-conduit » qui m'a permis d'affronter presque sereinement l'administration universitaire. Sans la formation que j'y ai suivie, je n'aurais probablement jamais pu faire cette thèse. À ce titre je remercie également l'ensemble des professeurs de ce Master, et plus particulièrement Guilhem Castagnos et Jean-Marc Couveignes qui m'ont enseigné les bases de la cryptographie.

Je tiens également à remercier Laila El Aimani pour m'avoir pris en stage chez Technicolor et m'avoir initié, avec toute sa gentillesse et sa bonne humeur, au monde de la recherche. Je remercie aussi Marc Joye, que j'ai rencontré là-bas, pour les discussions qu'il organisait et qui m'ont beaucoup appris.

Je suis extrêmement reconnaissant envers mes deux directeurs de thèse, David et Sébastien, pour m'avoir accepté en thèse et pour m'avoir offert une grande autonomie dans mon travail, tout en restant très disponibles. Les réunions que nous avons pu avoir ensemble ont chaque fois été très constructives et ont débouché sur bon nombre de résultats qui sont présentés dans ce mémoire.

J'ai passé la majeure partie de ces trois dernières années au sein de l'équipe NPS à Caen, où j'ai bénéficié d'un cadre très favorable. Je tiens d'ailleurs à remercier Jean-François pour avoir su maintenir une si bonne ambiance dans son équipe, permettant de travailler dans la bonne humeur. J'y ai particulièrement apprécié mes nombreuses discussions avec Jacques, plus efficace que DBLP pour retrouver une référence, que je remercie pour sa gestion du projet LYRICS (même si j'ai quelques doutes sur la validité de certains sondages Doodle...). J'ai également pu travailler avec Nicolas, qui m'a fait réaliser que la notion d'efficacité était très relative, et que sa perception différait fortement entre un cryptologue et un développeur. Je ne peux malheureusement pas citer chaque membre de l'équipe mais je leur suis à tous très reconnaissant pour ces trois années passées en leur compagnie.

L'autre aspect fantastique de cette équipe est le nombre de doctorants qui y évoluent et que j'ai pris beaucoup de plaisir à côtoyer. Je commencerai par Ghada qui a débuté le même jour que moi et qui a si souvent accepté (ou plutôt toléré) que je lui emprunte son encadrant. Il y a également Marie qui a partagé mon bureau (non Julien, je ne t'oublie pas, ton tour viendra...) pendant deux ans : comment vais-je faire pour recevoir mon xkcd quotidien désormais ? Là encore, je ne peux pas rentrer dans les détails pour tout le monde, mais merci à Amira, Chrystel, Julien, Kahina, Mohammed, Roch, Solenn et Yacine. Chacun d'entre eux peut (ou pourra) vous le confirmer, la rédaction d'un mémoire est une tâche difficile et parfois décourageante. Néanmoins, outre la nécessité de présenter mes travaux, celle-ci a pour ma part également été motivée par la nécessité de rétablir une vérité fondamentale : Julien, contrairement à ce que tu as pu écrire dans ton

mémoire, c'est bien TOI qui occupait MON bureau.

Même si, pour des raisons pratiques, je n'ai pu me rendre que trop rarement à l'ENS, j'ai à chaque fois été impressionné par la qualité des travaux qui y sont menés. Je n'ai malheureusement pas eu le temps de bien connaître les autres doctorants et chercheurs, mais j'avoue être très fier d'avoir pu évoluer au sein de cette institution.

Pour finir, je remercie toute ma famille (et belle-famille) et plus particulièrement ma femme qui m'a soutenu lors de mon changement de carrière et qui a accepté de me suivre en Normandie. Mais par dessus tout, je lui suis reconnaissant de m'avoir offert mes deux plus belles réussites de ces trois dernières années : mes filles Aurélie et Eugénie.

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 Présentation Générale

La cryptographie est une science ancienne (les premières traces connues datent du XVI^{e} siècle avant notre ère [Kah96]) répondant à un besoin probablement aussi vieux que l'écriture : celui de protéger l'information. Historiquement, la principale préoccupation était de cacher le contenu d'un message, le rendant inintelligible pour toute autre personne que ses destinataires. D'où le nom de cryptographie, du grec *kruptos* (« caché ») et *graphein* (« écrire »).

Cependant, ce problème, dit de la *confidentialité*, n'est pas le seul à se poser lors de la transmission ou le stockage d'un document. Il est également nécessaire de considérer celui de l'*intégrité*, consistant à s'assurer que le message n'a pas été modifié, et celui de l'*authenticité*, consistant à vérifier l'identité de l'émetteur.

1.1.1 Confidentialité

Discipline millénaire, la cryptographie n'a pourtant connu ses principales évolutions qu'au cours des dernières décennies. Elles sont essentiellement dues à l'avènement de l'informatique qui a bouleversé les besoins de sécurité traditionnels, mettant en évidence les limites des techniques existantes.

Pour le comprendre, il faut remonter aux premières constructions de l'histoire, qui visait à assurer la confidentialité des messages (on parle alors de système de *chiffrement*). Prenons l'exemple du chiffrement de César, devant son nom à Jules César qui l'utilisa pour ses correspondances secrètes. Ce système consiste à remplacer chaque lettre d'un message par celle qui le suit de 3 positions dans l'alphabet (*i.e.* A devient D, B devient E, etc...). Cet algorithme peut être généralisé pour n'importe quelle longueur de décalage (inférieure ou égale à 25 pour l'alphabet latin), cette valeur paramétrant le système étant appelée *clé*. La confidentialité ainsi obtenue repose alors sur la dissimulation de l'algorithme utilisé plutôt que sur celle de la clé. Un adversaire ayant identifié ce système de chiffrement n'aura en effet aucun mal à retrouver cette dernière en testant tous les décalages (et donc les clés) possibles.

Plus généralement, le fait de garder secret l'algorithme de chiffrement a longtemps été vu comme un moyen d'assurer la sécurité des transmissions. Cette approche, déjà critiquée par

Kerckhoff à la fin du XIX^{e} siècle, a été rendue obsolète par le développement des communications numériques. Celles-ci nécessitent en effet la définition de standards cryptographiques acceptés, et donc connus, par l'ensemble des membres du réseau. La sécurité ne peut alors reposer que sur la dissimulation de la clé secrète, qui doit donc appartenir à un espace suffisamment grand. Un exemple de tels standards est le DES [DES77] (pour *Data Encryption Standard*), développé par IBM en 1977, qui fut utilisé près de 20 ans jusqu'à ce que la taille de ses clés (56 bits) se révèle trop faible face à la puissance des ordinateurs modernes.

L'usage d'algorithmes publics, dont la robustesse peut être étudiée par l'ensemble de la communauté cryptographique, a incontestablement bénéficié à la sécurité des communications. Cependant, celle-ci ne repose alors que sur le secret des clés utilisées, ce qui met en évidence le problème de leur transmission. En effet, les systèmes de chiffrement ont presque toujours fonctionné sur le même principe : la connaissance nécessaire au chiffrement d'une donnée est la même que celle permettant son déchiffrement (on parle alors de chiffrement *symétrique* ou *à clé secrète*). Il est donc indispensable, pour les personnes souhaitant communiquer, de s'accorder sur une clé commune avant tout échange de messages. Traditionnellement, celle-ci était transférée aux destinataires des messages à l'aide d'un support physique (carnet, disquette, etc...). Malheureusement, cette solution est incompatible avec un monde moderne où il est par exemple possible d'effectuer des achats auprès de commerçants situés à des milliers de kilomètres de son domicile.

En proposant une réponse à ce problème d'échange de clés, Diffie et Hellman [DH76] ont véritablement révolutionné la cryptographie. Ils introduisirent en effet le concept de cryptographie à *clé publique* ou *asymétrique*, définissant deux clés différentes $(\mathsf{pk}, \mathsf{sk})$ pour un même utilisateur. La première, pk, peut être rendue publique (d'où son nom de clé publique) et permet à n'importe qui de chiffrer un message à destination de l'utilisateur, tandis que la seconde, sk (appelée clé secrète), ne sera connue que de ce dernier auquel elle permettra de déchiffrer les messages reçus. Le point essentiel est que la connaissance de la clé publique ne permet pas de déchiffrer un message ni même d'obtenir de l'information dessus. Si nous reprenons l'exemple d'un achat en ligne, le commerçant pourra simplement publier sur sa page internet sa clé publique qui sera utilisée par les acheteurs pour chiffrer les données sensibles qu'ils envoient (par exemple leur numéro de carte bancaire).

La cryptographie asymétrique ne doit cependant pas être opposée à sa variante symétrique mais plutôt être vue comme complémentaire. Elle offre en effet plus de possibilités mais souffre de performances plus faibles. En pratique, on utilise par exemple un algorithme de chiffrement asymétrique pour transmettre une donnée secrète qui servira à chiffrer, à l'aide d'un algorithme symétrique, des messages plus longs.

1.1.2 Intégrité et Authenticité

L'émergence des communications numériques n'a pas seulement eu des conséquences dans le domaine de la confidentialité. Elle a également créé un besoin de solutions cryptographiques assurant l'intégrité et l'authenticité. Jusque-là, ces propriétés étaient assurées par d'autres moyens, reposant sur l'utilisation d'un support physique pour le message.

Le contrôle de l'intégrité était par exemple simplifié par la difficulté de modifier un texte écrit sur une feuille de papier sans que cela soit détectable. Cette propriété n'est plus vraie pour une donnée numérique à laquelle on peut très simplement rajouter ou supprimer des éléments sans laisser de trace.

En ce qui concerne l'authenticité, la signature manuscrite, traditionnellement utilisée, a dû également être adaptée aux contraintes du monde numérique. En effet, s'il est difficile de récupérer la signature d'un document papier pour l'apposer sur un autre, il n'en va pas de même en informatique où le copier-coller est une opération aisée.

Tous ces challenges ont été relevés par la communauté cryptographique qui a proposé des solutions efficaces et sûres à ces problèmes. L'intégrité est par exemple assurée par l'usage de *fonctions*

de hachages cryptographiques telles que SHA-256 [SHA02] ou SHA-3 [SHA14]. La vérification de l'authenticité d'un document peut, elle, être réalisée à l'aide d'une signature numérique [DH76] que nous présentons dans la section 3.2 de ce mémoire.

Cependant les besoins de l'informatique vont bien au-delà des traditionnels problèmes de confidentialité, d'intégrité et d'authenticité. On peut notamment citer le problème du contrôle d'accès, où le fournisseur d'un service souhaitera identifier les utilisateurs y accédant. Mais il faut alors considérer sa contrepartie, à savoir le désir légitime de ces utilisateurs de protéger leur vie privée.

1.2 Cryptographie et Anonymat

Si les nouvelles technologies ont profondément modifié nos usages, elles ne présentent pas que des avantages pour leurs utilisateurs. Prenons l'exemple d'un résidant d'une grande ville française. Celui-ci dispose probablement d'une carte de transport lui permettant d'accéder au réseau de transport public sans avoir à acheter, puis conserver, plusieurs tickets. Par ailleurs, il utilise sûrement une carte bancaire, voire son téléphone mobile, pour régler ses courses ou une consultation chez son médecin. Ce mode de paiement lui permet en effet de ne plus avoir à transporter d'espèces et lui évite ainsi tous les problèmes associés (tels que celui de l'appoint).

L'adhésion massive du public à ces systèmes démontre bien tous les bénéfices qu'ils apportent à leurs utilisateurs. Cependant, ces technologies facilitent également leur traçage, sacrifiant ainsi l'anonymat offert par un ticket de bus ou par un billet de banque.

Il est surprenant de constater à quel point la valeur des informations révélées par un achat ou par la validation de sa carte de transport est minimisée par le grand public. Elles présentent pourtant un immense intérêt commercial comme l'a illustré le scandale de la vente des données des utilisateurs d'un système de paiement à Hong Kong [ZDN10]. Au-delà du simple fait qu'elles permettent de localiser précisément chaque individu, ces informations révèlent également ses habitudes, ce qui est suffisant pour en dresser un profil très précis. L'importance des dépenses médicales de l'utilisateur d'une carte bancaire peut par exemple permettre à sa banque d'évaluer son état de santé, ce qui pourra être pris en compte s'il demande un crédit.

Là encore, ces problématiques de sécurité mêlant authentification et anonymat se révèlent particulièrement complexes dans un environnement numérique. En effet, concevoir les équivalents électroniques d'un billet de banque ou d'une carte de transport est loin d'être trivial car la sécurité ne peut alors plus reposer sur l'utilisation d'un support physique difficilement réplicable.

Ce n'est pourtant pas impossible comme l'a montré Chaum [Cha82] en 1982 avec le concept de monnaie électronique ou en introduisant, avec Van Heyst [Cv91], la signature de groupe. Ces deux primitives sont des exemples, parmi d'autres, de la flexibilité de la cryptographie moderne, capable de répondre aux besoins de sécurité de bons nombres de cas d'usages, même les plus complexes.

Cependant ces résultats ne doivent pas faire oublier le problème principal de toute application pratique, à savoir l'efficacité. En effet, une solution cryptographique aura beau offrir les meilleures garanties de sécurité, elle ne sera pas utilisée si ses performances ne sont pas satisfaisantes.

1.3 Nos Contributions

Nous nous intéressons dans ce mémoire à différentes solutions permettant de rendre accessible au grand public toutes les possibilités offertes par la cryptographie en matière d'anonymat. À cette fin, nous étudierons deux pistes qui constituent les parties II et III de ce mémoire (la partie I présentant simplement les notions mathématiques et cryptographiques nécessaires à leur compréhension).

Tout d'abord, nous chercherons à concevoir de nouveaux protocoles répondant efficacement aux deux cas d'usage les plus fréquents, à savoir le paiement et le contrôle d'accès (en prenant l'exemple du transport public).

Paiement anonyme. En introduisant la notion de monnaie électronique, Chaum a cherché à reproduire, de manière électronique, le fonctionnement des espèces traditionnelles. Ce faisant, il a également transposé l'un de ses défauts, à savoir celui de l'appoint. Comme nous l'expliquerons dans le chapitre 4, ce problème ne peut pas être résolu aussi simplement que par un rendu de monnaie, comme ce serait le cas pour un billet de banque. Il est donc préférable, en pratique, d'utiliser une variante de la monnaie électronique, appelée *monnaie électronique divisible*, proposée par Okamoto et Ohta [OO92]. Celle-ci permet en effet de diviser une pièce en plusieurs parties qui pourront être dépensées séparément. Cependant, cette propriété s'est révélée extrêmement difficile à obtenir sans faire de compromis sur l'anonymat, comme l'illustre le faible nombre de constructions sûres de l'état de l'art. De plus, celles-ci souffrent d'une forte complexité, rendant peu probable leur utilisation en pratique.

Nous présenterons succinctement ces différentes solutions dans le chapitre 4 afin de mettre en évidence leurs limites intrinsèques, puis nous proposerons une nouvelle approche pour construire des schémas de monnaie électronique divisible. Ces travaux, présentés dans l'article *Divisible E-Cash Made Practical* [CPST15a] lors de la conférence PKC 2015, permettent aux utilisateurs de tels systèmes de bénéficier de performances remarquables, que ce soit lors d'un retrait de monnaie électronique ou lors d'une dépense.

Malheureusement, cette construction implique un nombre significatif de calculs pour superviser l'ensemble des dépenses du système, ce qui peut poser problème pour un service déployé à grande échelle. Nous décrirons donc, dans le même chapitre, une solution évitant cet écueil qui a fait l'objet de l'article *Scalable Divisible E-Cash* [CPST15b], publié dans la conférence ACNS 2015.

Authentification anonyme. Les problèmes de respect de la vie privée ne concernent pas le seul cas du paiement et peuvent se retrouver dans d'autres situations de la vie courante. Ces dernières étant toutes différentes, il est souvent préférable de proposer des solutions cryptographiques qui leur sont spécifiquement dédiées. En effet, une solution générique peut difficilement respecter les exigences de sécurité propres à chaque contexte tout en offrant une efficacité optimale.

Nous proposerons donc, dans le chapitre 5, une nouvelle primitive cryptographique adaptée au cas du transport public, offrant plus de flexibilité dans les procédures de levée d'anonymat que les signatures de groupes ou certaines de leurs variantes. Celle-ci peut être instanciée efficacement, comme nous le montrerons dans ce même chapitre. Ces travaux ont été présentés dans l'article *Direct Anonymous Attestation with Dependent Basename Opening* [DLST14] lors de la conférence CANS 2014.

La conception de nouveaux protocoles n'est malheureusement pas toujours suffisante pour satisfaire des contraintes de performances toujours plus fortes. Nous étudierons donc, dans la dernière partie de ce mémoire, différentes techniques pour optimiser ces protocoles. Celles-ci correspondent à deux approches que nous décrivons ci-dessous.

Délégation de Calculs. L'efficacité d'une construction est un concept très relatif, dépendant notamment du type de périphérique sur laquelle elle sera implémentée. En effet, un protocole qui s'exécute « efficacement » sur un ordinateur récent pourra par exemple être hors de portée d'une carte à puce. Cet écart de performances entre les différents périphériques peut être une opportunité lorsqu'ils sont connectés. Il est alors possible, pour le dispositif le plus faible, de tirer parti de la puissance de calcul des autres entités.

Ce problème de *délégation de calculs* sera étudié dans ce mémoire à deux niveaux. Dans le chapitre 6, nous le considèrerons à l'échelle d'un algorithme. Cela consiste à décider, pour toutes les opérations composant ce dernier, lesquelles peuvent être effectuées par un délégataire.

Malheureusement, cette décision est loin d'être simple et doit prendre en compte des problèmes de sécurité rendus particulièrement complexes par l'introduction dans le système d'une nouvelle entité potentiellement corrompue. Si ce processus de décision peut être accéléré, comme nous l'avons expliqué dans l'article *Toward Generic Method for Server-Aided Cryptography* [CCD+13], certaines étapes nécessitent un traitement propre à chaque cas.

Afin d'augmenter la portée de ces travaux, il peut donc être intéressant de chercher à déléguer des algorithmes souvent utilisés dans des constructions cryptographiques. La version coopérative obtenue peut alors servir à de multiples reprises. Nous verrons par exemple que les protocoles anonymes sont souvent construits à l'aide de « briques cryptographiques » telles que la signature numérique ou la preuve de connaissance (que nous définirons toutes les deux dans le chapitre 3). Cette dernière étant souvent coûteuse, nous proposerons un moyen de la déléguer qui pourra servir à accélérer l'exécution de l'ensemble des constructions les utilisant. Cette solution a été présentée dans l'article *Efficent Delegation of Zero-Knowledge Proofs of Knowledge in a Pairing-Friendly Setting* [CPS14], lors de la conférence PKC 2014.

Cette approche qui considère les opérations mathématiques comme les atomes d'un algorithme à répartir entre différentes entités n'est cependant pas complète. En effet, il est également possible de considérer la délégation des opérations elles-mêmes. Celle-ci se justifie en particulier pour les calculs mathématiques coûteux tels que l'exponentiation ou le couplage bilinéaire (défini dans le chapitre 2) qui sont fréquemment utilisés en cryptographie. Nous présenterons donc dans le chapitre 7 un état de l'art des principaux travaux sur ce sujet ainsi que les résultats que nous avons obtenu dans le cas du couplage bilinéaire et qui ont fait l'objet de l'article *Delegating a Pairing can be Both Secure and Efficient*, publié dans la conférence ACNS 2014.

Instanciation Efficace de Briques Cryptographiques. Une autre approche pour améliorer l'efficacité des protocoles consiste à concevoir de nouveaux schémas instanciant les briques cryptographiques usuelles. Nous prendrons l'exemple dans le chapitre 8 de la signature numérique. Nous y décrirons une nouvelle construction satisfaisant les propriétés généralement attendues de cette primitive lorsqu'elle est utilisée comme brique cryptographique, mais avec une bien meilleure efficacité que les solutions existantes. Elle peut donc être substituée à ces dernières dans la majorité des protocoles protégeant la vie privée de leurs utilisateurs, les rendant ainsi plus performants. Ces travaux sont issus de l'article *Short Randomizable Signature* [PS15].

Première partie

Préliminaires

Chapitre 2

Outils Mathématiques

Sommaire

Nous rappelons dans ce chapitre les notions mathématiques indispensables à la compréhension de ce mémoire. Nous commencerons par présenter les groupes algébriques et les corps et donnerons quelques exemples de ceux fréquemment utilisés en cryptographie. Nous introduirons ensuite les groupes bilinéaires et décrirons les problèmes considérés comme difficiles dans ceux-ci.

2.1 Structures Algébriques

2.1.1 Groupe

Définition 1 (Groupe)**.** Un *groupe* $(\mathbb{G}, \cdot)$ est un ensemble $\mathbb{G}$ muni d'une loi de composition interne $\cdot : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ vérifiant les propriétés suivantes.

- *Associativité :* $\forall (x, y, z) \in \mathbb{G}^3$, $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.
- *Existence d'un élément neutre :* $\exists\ e \in \mathbb{G}$ tel que $\forall x \in \mathbb{G}$, $x \cdot e = e \cdot x = x$.
- *Existence d'un inverse :* $\forall\ x \in \mathbb{G}$, il existe un élément $y \in \mathbb{G}$ tel que $x \cdot y = y \cdot x = e$.

Nous adopterons dans ce mémoire la notation multiplicative pour la loi de groupe. Par conséquent, l'inverse d'un élément $x \in \mathbb{G}$ sera noté x^{-1} et l'élément neutre $1_{\mathbb{G}}$.

Définition 2. Soit $(\mathbb{G}, \cdot)$ un groupe et $\mathbb{H}$ un sous-ensemble de $\mathbb{G}$. $(\mathbb{H}, \cdot)$ est un *sous-groupe* de $(\mathbb{G}, \cdot)$ si les trois conditions suivantes sont satisfaites :

- $1_{\mathbb{G}} \in \mathbb{H}$;
- $\forall\ (x, y) \in \mathbb{H}^2$, $x \cdot y \in \mathbb{H}$;
- $\forall\ x \in \mathbb{H}$, $x^{-1} \in \mathbb{H}$.

Un sous-groupe est donc en particulier un groupe. Pour simplifier l'écriture, nous désignerons dans le reste de ce mémoire le groupe $(\mathbb{G}, \cdot)$ par $\mathbb{G}$.

Définition 3.

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

- *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 le *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 $\mathbb{G}$.
- *Ordre d'un groupe :* le cardinal de $\mathbb{G}$, que l'on notera $|\mathbb{G}|$ est appelé *ordre* du groupe $\mathbb{G}$. Un groupe est dit *fini* si son ordre est fini.
- *Ordre d'un élément :* l'*ordre d'un élément* $x \in \mathbb{G}$ est l'ordre du sous-groupe $< x >$ qu'il engendre.

Tous les groupes considérés dans ce mémoire seront commutatifs et finis. Le théorème ci-dessous présente un résultat majeur pour les groupes finis.

Théorème 4 (Lagrange)**.** *Soient $\mathbb{G}$ un groupe fini et $\mathbb{H}$ un sous-groupe de $\mathbb{G}$. L'ordre de $\mathbb{H}$ divise celui de $\mathbb{G}$.*

Ce théorème implique en particulier que l'ordre d'un élément divise celui du groupe. De plus, il permet de montrer que tout groupe d'ordre premier est cyclique et que chacun de ses éléments $x \neq 1_G$ en est un générateur. Ce dernier résultat sera implicitement utilisé dans les protocoles cryptographiques où il est souvent nécessaire de sélectionner des générateurs d'un groupe d'ordre premier.

2.1.2 Corps

Définition 5 (Anneau)**.** Un *anneau* $(\mathbb{A}, +, \cdot)$ est un ensemble $\mathbb{A}$ muni de deux lois de composition interne $+ : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ et $\cdot : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ vérifiant les propriétés suivantes :

- $(\mathbb{A}, +)$ est un groupe commutatif
- la loi $\cdot$ est associative
- la loi . est distributive par rapport à $+$, *i.e.* $\forall\ (x, y, z) \in \mathbb{A}^3$, $x \cdot (y + z) = x \cdot y + x \cdot z$

Un anneau est dit *unitaire* s'il contient un élément neutre $1_{\mathbb{A}}$ pour la loi $\cdot$ différent de l'élément neutre pour la loi $+$ que l'on notera $0_{\mathbb{A}}$.

Un exemple célèbre d'anneau unitaire est l'anneau des entiers relatifs $\mathbb{Z}$. Dans ce mémoire, nous ne manipulerons pas directement des anneaux unitaires mais un de leurs sous-ensembles, appelé *corps*.

Définition 6 (Corps)**.** Un *corps* $\mathbb{K}$ est un anneau unitaire dont tous les éléments différent de $0_{\mathbb{K}}$ sont inversibles pour la loi $\cdot$. Un corps est dit *fini* si son cardinal est fini.

Exemple 1: L'ensemble des nombres réels $\mathbb{R}$ muni de l'addition et de la multiplication usuelles est un corps infini. L'ensemble $\{0, 1, \ldots, p-1\}$ des entiers inférieurs à un nombre premier p muni de l'addition et de la multiplication modulo p est un corps fini qui sera noté $\mathbb{Z}_p$.

Seuls les corps finis seront utilisés dans ce mémoire. Nous détaillons ci-dessous quelques résultats majeurs les concernant. Ces résultats seront implicitement utilisés dans les prochains chapitres.

Théorème 7.

1. *(Wedderburn) : tout corps fini est commutatif.*
2. *Le cardinal d'un corps fini est la puissance d'un nombre premier p, appelé caractéristique. Un tel corps contient alors un sous-corps isomorphe à $\mathbb{Z}_p$.*
3. *Réciproquement, pour tout nombre premier p et tout entier positif n, il existe un unique (à isomorphisme près) corps $\mathbb{K}$ de cardinal p^n. Un tel corps sera noté sans ambiguïté $\mathbb{F}_{p^n}$.*
4. *Soit $\mathbb{K}$ un corps fini. L'ensemble $\mathbb{K} \setminus \{0_{\mathbb{K}}\}$, noté $\mathbb{K}^*$, muni de la loi $\cdot$ est un groupe cyclique, appelé groupe multiplicatif de $\mathbb{K}$.*

Les deuxième et troisième points impliquent donc que $\mathbb{F}_p = \mathbb{Z}_p$ lorsque p est premier. En revanche, il est important de noter que $\mathbb{F}_{p^n} \neq \mathbb{Z}_{p^n}$ lorsque $n > 1$, $\mathbb{Z}_{p^n}$ n'étant pas un corps.

2.1.3 Exemples d'utilisation en cryptographie

Les groupes sont fréquemment utilisés en cryptographie à clé publique car ils donnent lieu à des problèmes algorithmiques permettant de garantir la sécurité des protocoles. Nous détaillerons dans la section 2.2.3 l'ensemble des problèmes qui seront considérés dans ce mémoire. Cependant, ceux-ci reposent sur le *problème du logarithme discret en base x* qui consiste, étant donnés $x \in \mathbb{G}$ et $y \in \langle x \rangle$, à trouver un entier $n \in \mathbb{N}$ tel que $y = x^n$. Si l'ordre de x est petit, résoudre le problème du logarithme discret en cette base devient très facile. En pratique, on cherchera donc à choisir une base qui soit d'ordre maximal au sein du groupe. Cela explique pourquoi les groupes d'ordre premier sont souvent privilégiés en cryptographie. En effet, tous leurs éléments, à part l'élément neutre, sont d'ordre maximal et peuvent donc être choisis comme base. L'ordre du groupe n'est cependant pas le seul critère, comme l'illustrent les exemples ci-dessous.

Sous-groupe de $\mathbb{F}_p^*$

Soit p un nombre premier, le théorème 7 assure que le groupe multiplicatif $\mathbb{F}_p^*$ du corps $\mathbb{F}_p$ est cyclique à $p-1$ éléments. Comme expliqué ci-dessus, on souhaite en général utiliser des groupes d'ordre premier. On choisira donc $\mathbb{G}$ comme le sous-groupe d'ordre q de $\mathbb{F}_p^*$ pour un certain nombre premier q divisant $p-1$. En pratique, il est aujourd'hui recommandé [ANS13] d'utiliser un premier p de taille 3072 bits admettant un premier q d'au moins 200 bits comme diviseur de $p-1$.

Bien que très souvent utilisés au début de la cryptographie asymétrique, ces groupes sont progressivement abandonnés en raison de leur taille de plus en plus importante. Celle-ci est due à la vulnérabilité de ces groupes à la méthode dite de calcul d'index que l'on doit notamment à Adleman [Adl79] et qui permet de retrouver le logarithme discret d'un élément beaucoup plus efficacement qu'en utilisant les méthodes génériques (celles qui fonctionnent pour n'importe quel groupe).

Groupe de points d'une courbe elliptique

Il existe de nombreuses façons de définir une courbe elliptique qui font bien souvent appel à des notions de mathématiques très avancées. Nous choisissons ici de présenter une définition plus simple, mais moins complète, et renvoyons le lecteur vers l'ouvrage de Cohen *et al* [CFA+12] pour une étude plus approfondie du sujet.

Définition 8. Soit $p > 3$ un nombre premier et $q = p^n$ pour un certain $n > 0$. Une *courbe elliptique* E sur $\mathbb{F}_q$ est une équation de la forme :

$$Y^2 = X^3 + a \cdot X + b$$

avec $a, b \in \mathbb{F}_q$. Une courbe est dite *singulière* si $4a^3 + 27b^2 = 0$ et *non-singulière* autrement.

Il est également possible de définir des courbes elliptiques sur des corps de caractéristiques 2 ou 3 mais l'équation devient alors un peu plus complexe. Nous n'utiliserons dans ce mémoire que des courbes non-singulières sur des corps de grande caractéristique.

Définition 9. Soit E une courbe elliptique sur $\mathbb{F}_q$. Pour tout $k \geq 1$, on définit $E(\mathbb{F}_{q^k})$ comme l'ensemble des solutions (x, y) de E dans $\mathbb{F}_{q^k}$ auquel on ajoute un point spécial, noté P_∞, appelé *point à l'infini*.

Pour obtenir un groupe, il reste donc à munir $E(\mathbb{F}_{q^k})$ d'une loi de composition interne. De manière informelle, celle-ci, notée $+$ et d'élément neutre P_∞ sera définie comme suit.

– Si $P = (x_P, y_P) \in E(\mathbb{F}_{q^k})$, son inverse, noté $-P$ a pour coordonnées $(x_P, -y_P)$.

- La tangente à la courbe au point $P \in E(\mathbb{F}_{q^k})$ coupe celle-ci en un autre point $R = (x_R, y_R) \in E(\mathbb{F}_{q^k})$. La somme $[2]P = P + P$ est alors définie comme $-R$.
- La droite passant par deux points distincts $P \in E(\mathbb{F}_{q^k})$ et $Q \in E(\mathbb{F}_{q^k})$ coupe la courbe en un troisième point $R = (x_R, y_R) \in E(\mathbb{F}_{q^k})$. La somme $P + Q$ est alors définie comme $-R$ (voir figure 2.1).

Il est possible de montrer que cette loi vérifie toutes les propriétés requises, notamment l'associativité. Par ailleurs, la commutativité du groupe $(E(\mathbb{F}_{q^k}), +)$ est immédiate par construction.

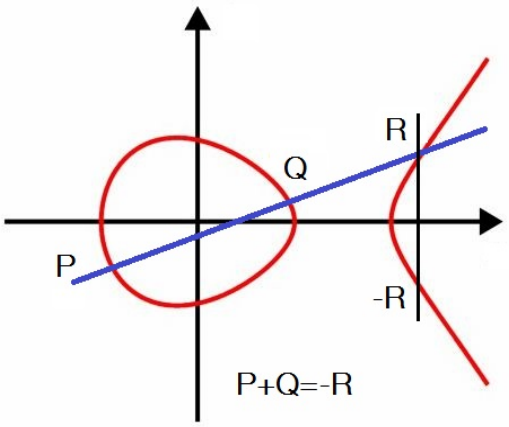


Figure 2.1 – Addition de points distincts sur courbe elliptique.

L'un des principaux avantages des courbes elliptiques est que, contrairement aux sous-groupes de $\mathbb{F}_p^*$, on ne connait pas d'algorithme plus efficace que les méthodes génériques (c'est-à-dire celles qui fonctionnent pour n'importe quel groupe) pour résoudre le problème du logarithme discret. Cela permet d'utiliser, pour un même niveau de sécurité, des paramètres de tailles nettement plus faibles (un nombre q de 256 bits suffit [ANS13] contre 3072 bits pour les sous-groupes de $\mathbb{F}_p^*$).

Un autre avantage de ces courbes est qu'elles permettent de construire efficacement les groupes bilinéaires que l'on définit ci-dessous.

2.2 Couplages

Bien que différents, les groupes de points d'une courbe elliptique et les groupes multiplicatifs d'un corps fini sont en fait reliés par une application appelée *couplage*. Celle-ci, prenant en entrée deux points d'une courbe elliptique E sur $\mathbb{F}_p$ et à valeurs dans $\mathbb{F}_{p^k}$ pour un certain entier k, permet de transférer des relations de $E(\mathbb{F}_{p^k})$ dans $\mathbb{F}_{p^k}^*$. En 1991, Menezes, Vanstone et Okamoto [MVO91] l'ont utilisée pour résoudre le problème du logarithme discret dans certaines familles de courbes elliptiques pour lesquelles k est petit. En effet, le couplage permet de transformer une instance du problème du logarithme discret dans $E(\mathbb{F}_{p^k})$ en une instance dans un sous-groupe de $\mathbb{F}_{p^k}^*$, où il est plus simple à résoudre.

Ce n'est que plusieurs années plus tard que Joux [Jou00] montra que les couplages pouvaient être utilisés à d'autres fins que la cryptanalyse. En effet, la bilinéarité de telles applications offre une multitude de possibilités et a servi dans de nombreuses constructions cryptographiques telles que le chiffrement basé sur l'identité [BF01], la signature numérique [BLS01] ou la monnaie électronique [CPST15a].

Les primitives cryptographiques à base de couplages utilisent la notion de *groupes bilinéaires* qui permet d'éviter de rentrer dans les détails de la construction mathématiques des couplages.

Nous suivrons la même approche pour la plupart des schémas décrits dans ce mémoire et commençons donc par donner la définition de tels groupes. Cependant, le protocole de délégation de couplages décrit dans le chapitre 7 nécessitera quelques détails sur leur construction que nous fournissons dans ce qui suit. Pour finir nous présenterons une liste de problèmes considérés comme difficiles dans de tels groupes.

2.2.1 Groupes Bilinéaires

Définition 10 (Groupes Bilinéaires)**.** Les *groupes bilinéaires* sont un ensemble de trois groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ d'ordre p premier muni d'une application $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ vérifiant 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.

La notion de « calculable efficacement » sera définie ci-dessous. Cependant elle correspond intuitivement à l'existence d'un algorithme dont le temps d'exécution est polynomial en la taille de ses entrées. Cette condition est importante car elle permet d'exclure certaines constructions irréalistes en pratique. Par exemple, considérons un groupe $\mathbb{G}$ d'ordre premier p et $g \in \mathbb{G}$ un générateur de celui-ci. Pour tout $(h, f) \in \mathbb{G}^2$, il existe $(a, b) \in \mathbb{Z}_p^2$ tel que $(h, f) = (g^a, g^b)$. Définissons alors e comme l'application $\mathbb{G} \times \mathbb{G} \to \mathbb{G}$ définie par $e(h, f) = g^{a \cdot b}$. Cette application est donc bien définie et vérifie les deux premières propriétés. Cependant, sans hypothèse supplémentaire sur $\mathbb{G}$, le calcul de e nécessite de retrouver les logarithmes discrets de h et f en base g, ce qui est censé être un problème difficile (sinon $\mathbb{G}$ ne serait pas utilisé en cryptographie). Par conséquent, e ne vérifie pas la dernière propriété et ne permet donc pas de définir des groupes bilinéaires.

Remarque 1. Il est également possible de définir des groupes bilinéaires d'ordre N *composite*, *i.e.* non premier. De tels groupes, introduits en cryptographie par Boneh, Goh et Nissim [BGN05], proposent des fonctionnalités intéressantes mais souffrent d'une taille très importante. De plus, la complexité des opérations dans ces groupes est nettement plus forte que dans les groupes d'ordre premier, comme l'a montré Guillevic [Gui13].

En pratique, les groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ sont construits à partir d'une courbe elliptique tandis que $\mathbb{G}_T$ est un sous-groupe d'un corps fini. Le choix de la courbe est déterminant pour certaines propriétés de $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ notamment l'existence (ou non) de relations entre ces groupes. Galbraith, Paterson et Smart [GPS08] en ont fait un critère permettant de diviser l'ensemble des groupes bilinéaires en 3 types.

- *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 1 isomorphisme efficacement calculable $\phi_2 : \mathbb{G}_2 \to \mathbb{G}_1$ mais aucun dans le sens inverse ;
- *type 3 :* il n'existe aucun isomorphisme efficacement calculable entre $\mathbb{G}_1$ et $\mathbb{G}_2$.

Là encore, la notion de « calculable efficacement » est importante. En effet, comme $\mathbb{G}_1$ et $\mathbb{G}_2$ sont deux groupes de même ordre premier, ils sont isomorphes. Le point important pour la sécurité des schémas cryptographiques utilisant les couplages de types 2 ou 3 est qu'il n'existe pas d'algorithme efficace permettant de calculer ces isomorphismes.

Remarque 2. Les groupes de type 1 sont souvent définis comme des groupes tels que $\mathbb{G}_1 = \mathbb{G}_2$. Il est important de noter qu'en pratique ces deux groupes ne sont pas nécessairement identiques et qu'il s'agit d'un abus de notation justifié par l'existence de ces isomorphismes permettant de transposer facilement un élément d'un groupe à l'autre.

Les premières constructions cryptographiques à base de couplages ont très majoritairement utilisé des groupes de type 1. Cependant, la présence de ces deux isomorphismes donnent plus de

possibilités à un adversaire attaquant ces constructions. En pratique, cela se traduit par le fait que certains problèmes algorithmiques jugés difficiles pour les groupes de types 2 ou 3 se révèlent faciles lorsque des groupes de type 1 sont utilisés. Cela explique en partie l'abandon progressif de ces derniers au profit des groupes de type 3 qui offrent par ailleurs une meilleure efficacité.

Les groupes (ou couplages) de type 1 sont en général dits *symétriques* par opposition à ceux de type 2 ou 3 dits *asymétriques*. Concernant ces derniers, les groupes de type 2 ne sont pratiquement plus utilisés car on ne leur connait, pour l'instant, aucun avantage par rapport à ceux de type 3, comme expliqué dans [CM11].

Pour la grande majorité des protocoles décrits dans ce mémoire, il ne sera pas nécessaire de savoir comment sont construits ces groupes mais simplement de savoir qu'ils peuvent être générés très efficacement. Cependant, le protocole décrit au chapitre 7 nécessite de connaître quelques détails relatifs à leur construction que nous décrivons ci-dessous.

2.2.2 Calcul de Couplages

De même que pour les courbes elliptiques, définir les couplages nécessite d'introduire des notions complexes de mathématiques qui sortent du cadre de ce mémoire. Malheureusement, contrairement aux courbes elliptiques dont la loi de groupe peut être décrite géométriquement, il est difficile d'expliquer de manière intuitive ce qu'est un couplage. Nous nous contenterons donc d'introduire seulement ce qui est nécessaire et renvoyons à nouveau le lecteur vers [CFA+12] pour plus de détails.

En cryptographie, les couplages sont construits essentiellement à partir de courbes elliptiques ou hyperelliptiques. D'autres techniques, basées sur les réseaux euclidiens, ont été proposées [GGH13, CLT13, LSS14] et permettent de construire des *applications multilinéaires* dont les couplages sont un cas particulier. Malheureusement, celles-ci souffrent d'une complexité nettement supérieure aux constructions basées sur courbes elliptiques et leur sécurité a été récemment mise en cause [CHL+14, GHMS14, CLT14a]. Nous nous intéresserons ici uniquement aux couplages basés sur courbes elliptiques.

Définition 11. Soient E une courbe elliptique sur $\mathbb{F}_q$ et $\mathbb{G}$ un sous-groupe de $E(\mathbb{F}_q)$ d'ordre p premier tel que $p \nmid q$. Le *degré de plongement* de E est le plus petit entier k tel que $p|(q^k - 1)$.

Il existe deux types de couplages : le *couplage de Weil* et le *couplage de Tate*. Si le premier a été principalement utilisé au début de la cryptographie basée sur les groupes bilinéaires, on lui préfère aujourd'hui le couplage de Tate ou l'une de ses nombreuses variantes [BGOS07, HSV06, Ver10]. Cet algorithme définit $\mathbb{G}_1$ comme un sous groupe de $E(\mathbb{F}_q)$ et $\mathbb{G}_2$ comme un sous groupe de $E(\mathbb{F}_{q^k})$, ou inversement. Le groupe $\mathbb{G}_T$ est défini comme le sous groupe de $\mathbb{F}^*_{q^k}$ d'ordre p (celui-ci existe d'après la définition du degré de plongement). Le couplage de Tate peut être décomposé en deux parties : l'*algorithme de Miller*, qui diffère selon la variante utilisée, et l'*exponentiation finale*, qui élève la sortie de l'algorithme de Miller à la puissance $\frac{q^k-1}{p}$.

En pratique, les trois paramètres majeurs du couplage sont donc q, p et k. Cependant, plusieurs facteurs doivent être pris en considération lors de leur choix. Tout d'abord, le théorème de Hasse montre que le cardinal de $E(\mathbb{F}(q))$ est d'un ordre de grandeur comparable à celui de q (on notera $|E(\mathbb{F}(q))| \approx q$). Afin d'optimiser les paramètres du couplage on cherchera donc à choisir une courbe E telle que $E(\mathbb{F}_q)$ contienne un sous-groupe d'ordre $p \approx q$ avec idéalement $p = |E(\mathbb{F}(q))|$. Cependant, pour des valeurs aléatoires de p et q avec $p \approx q$ on obtient un degré de plongement k proche de p [BK98] et donc un groupe $\mathbb{G}_T$ dont les éléments ont une taille de 2^{256} bits, ce qui rend inconcevable tout calcul. Il est donc important de choisir p et q de sorte que k reste assez petit. Malheureusement, un k trop petit pose également problème car cela risque de rendre facile le problème du logarithme discret dans $\mathbb{F}^*_{q^k}$ et donc de rendre nos groupes vulnérables à l'attaque due à Menezes, Okamoto et Vanstone décrite ci-dessus.

Le choix de ces paramètres dépend aujourd'hui principalement du type des groupes bilinéaires. Les groupes symétriques sont instanciés à l'aide de courbes *supersingulières*, *i.e.* telles que $|E(\mathbb{F}_q)| = q + 1$, dont le degré de plongement est 2 lorsque q est de grande caractéristique. Afin d'atteindre la taille recommandée de 3072 bits pour $\mathbb{F}^*_{q^2}$, il est donc nécessaire de prendre $q \geq 2^{1536}$, ce qui annule en grande partie l'intérêt des courbes elliptiques. Les groupes asymétriques offrent pour leurs parts plus de flexibilité dans le choix de ces paramètres. En pratique, on utilise essentiellement les courbes proposées par Barreto et Naehrig [BN06] qui proposent des paramètres optimaux au vu des recommandations de sécurité actuelles. En effet celles-ci permettent de choisir $p = |E(\mathbb{F}_q)| \approx q \approx 2^{256}$ et $k = 12$ ce qui donne $|\mathbb{F}^*_{q^k}| \approx 2^{3072}$ et assure ainsi un niveau de sécurité suffisant tant sur la courbe elliptique que sur le corps fini. En particulier, il est important de noter que l'on obtient un groupe $\mathbb{G}_1 = E(\mathbb{F}_q)$ de taille minimale par rapport au niveau de sécurité souhaité et que par conséquent la cryptographie à base de couplages n'est pas nécessairement beaucoup plus couteuse que celle à base de courbes elliptiques du moment que le nombre d'opérations dans $\mathbb{G}_2$ et $\mathbb{G}_T$ est limité.

2.2.3 Problèmes Difficiles

Idéalement, les protocoles cryptographiques devraient être sûrs même en présence d'un adversaire disposant d'une puissance de calcul illimitée. Le chiffrement de Vernam est un exemple de tels protocoles que l'ont dit sûrs au sens de la théorie de l'information. Malheureusement, les schémas vérifiant cette propriété souffrent d'une complexité qui empêche leur utilisation à grande échelle. Dans le cas du chiffrement, il peut par exemple être prouvé que la taille de la clé utilisée doit être supérieure à la longueur combinée des messages que celle-ci chiffrera. La cryptographie moderne s'est donc orientée vers une notion plus faible, mais plus réaliste, appelée *sécurité calculatoire*. Celle-ci considère un schéma comme sûr si, *sous certaines hypothèses*, aucun adversaire disposant d'une puissance de calcul *raisonnable* n'est capable d'en casser la sécurité avec une probabilité *non négligeable*. Définir ces hypothèses ainsi que les notions de ≪raisonnable≫ et de ≪non négligeable≫ est l'objectif de cette section.

L'un des assouplissements apporté par la sécurité calculatoire est donc le fait qu'elle considère désormais des adversaires disposant d'une puissance de calcul limitée. En pratique on pourra par exemple considérer un adversaire disposant d'une puissance de calcul équivalente à celle d'un supercalculateur fonctionnant durant des millénaires. L'utilisateur du système cryptographique sera alors rassuré par le fait que ses secrets ne seront révélés qu'après plusieurs générations. Formellement, cela se traduit par la notion d'*algorithme probabiliste polynomial* ainsi défini :

Définition 12. Un algorithme A est dit *polynomial* (ou s'exécutant en temps polynomial) s'il existe un polynôme $p(.)$ tel que, pour toute entrée $x \in \{0,1\}^*$, le temps d'exécution de $A(t)$ est majoré par $p(|x|)$ (où $|x|$ désigne la longueur de x).

Un algorithme A est dit *probabiliste* s'il a accès à une source d'aléa générant de manière indépendante des bits uniformément distribués sur $\{0,1\}$.

Une autre nouveauté de la sécurité calculatoire est qu'elle considère comme sûrs des schémas pouvant pourtant être cassés avec une probabilité infime. Cela est défini formellement par la notion de *fonction négligeable*.

Définition 13. Une fonction f est *négligeable* si pour tout polynôme $p(.)$, il existe un entier N tel que, $\forall n \geq N$, on a $f(n) < \frac{1}{p(n)}$.

Finalement, la dernière concession faite par la sécurité calculatoire est que la sécurité des constructions cryptographiques repose maintenant sur des hypothèses. Celles-ci consistent à supposer qu'un problème donné est *difficile*, c'est-à-dire que la probabilité de succès d'un algorithme probabiliste polynomial le résolvant est majoré par une fonction négligeable.

Cependant, cette définition de la difficulté reste asymptotique et ne permet pas de fixer directement la taille des paramètres à choisir. En pratique ceux-ci seront choisis en fonction de l'efficacité des meilleurs algorithmes permettant de résoudre le problème considéré ainsi que de la puissance de calcul qu'un adversaire peut avoir à sa disposition. Par exemple, le problème du logarithme discret en base x mentionné au chapitre 2.1.3 est aujourd'hui considéré comme difficile dans le groupe $\mathbb{F}_q^*$ pour $q \geq 2^{3072}$ [ANS13].

Nous présentons maintenant les différents problèmes relatifs aux groupes bilinéaires que nous utiliserons dans ce mémoire. Nous commençons cependant par présenter le problème du logarithme discret pour les groupes d'ordre premier. Ce problème est fondamental car tous les problèmes que nous allons introduire reposent sur lui. Plus précisément, si le problème du logarithme discret dans les groupes bilinéaires se révélait facile, alors il en serait de même pour tous les autres.

Définition 14. Soit $\mathbb{G}$ un groupe d'ordre premier. Le *problème du logarithme discret* (DL) consiste, étant donné $(g, h) \in \mathbb{G}^2$, à calculer $x \in \mathbb{Z}_p$ tel que $h = g^x$.

Nous avions introduit précédemment le problème du logarithme discret en une certaine base. Comme nous nous sommes restreints ici au cas des groupes d'ordres premiers, tous leurs éléments, sauf l'élément neutre, en sont des générateurs. Par conséquent, la difficulté du problème DL est la même quelle que soit la base considérée, ce qui nous permet d'omettre cette dernière.

Définition 15. Soit $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires. Le *problème du logarithme discret symétrique* (SDL) consiste, étant donné $(g, g^x, \widetilde{g}, \widetilde{g}^x) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2$, à calculer $x \in \mathbb{Z}_p$.

Ce problème, formalisé dans [BCN+10], est une extension du problème du logarithme discret aux groupes bilinéaires. Il est utilisé de manière implicite par de nombreux protocoles, en particulier ceux définis pour des couplages symétriques.

Définition 16. Soit $\mathbb{G}$ un groupe d'ordre premier. Le *problème calculatoire Diffie-Hellman* (CDH) consiste, étant donné $(g, g^a, g^b) \in \mathbb{G}^3$, à calculer $g^{a \cdot b}$.

Ce problème, introduit par Diffie et Hellman [DH76] pour évaluer la sécurité de leur protocole d'échange de clés est aujourd'hui considéré comme classique. Il existe une variante *décisionnelle* de ce problème que nous décrivons ci-dessous.

Définition 17. Soit $\mathbb{G}$ un groupe d'ordre premier. Le *problème décisionnel Diffie-Hellman* (DDH) consiste, étant donné $(g, g^a, g^b, g^z) \in \mathbb{G}^4$, à décider si $z = a \cdot b$ ou si z est un scalaire aléatoire.

On peut noter que le problème DDH est plus facile que le problème CDH, c'est-à-dire qu'un algorithme capable de résoudre ce dernier pourra être utilisé pour résoudre le premier. Il existe d'ailleurs des groupes pour lesquels le problème DDH est facile alors que CDH reste difficile. Les groupes bilinéaires de type 1 en sont un exemple car la propriété $z = a \cdot b$ peut être testée à l'aide du couplage : $e(g^a, g^b) \stackrel{?}{=} e(g^z, g)$. Cependant, le problème DDH reste difficile (au moins dans $\mathbb{G}_1$) pour les autres types de couplage car leur asymétrie empêche de réaliser le même test. On définit alors le problème XDH dans $\mathbb{G}_i$ pour $i \in \{1, 2\}$ comme étant le problème DDH dans $\mathbb{G}_i$ (on utilise une notation différente car l'existence d'un couplage offre théoriquement plus de pouvoir à l'adversaire et définit donc un problème différent). L'hypothèse SXDH suppose que le problème XDH est difficile à la fois dans $\mathbb{G}_1$ et dans $\mathbb{G}_2$, ce qui est le cas pour les couplages de type 3.

Définition 18. Soit $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires. Le problème DBDH consiste, étant donnés $(g, g^a, g^b, g^c, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^b, \widetilde{g}^c) \in \mathbb{G}_1^4 \times \mathbb{G}_2^4$ et $e(g, \widetilde{g})^z \in \mathbb{G}_T$, à décider si $z = a \cdot b \cdot c$ ou si z est aléatoire.

Ce problème, formalisé sous un autre nom dans [BF01], fut utilisé pour la première fois (bien que de manière implicite) dans l'échange de clés tripartite proposé par Joux [Jou00]. Il s'agit d'une extension assez naturelle du problème DDH aux groupes bilinéaires.

Définition 19. Soit $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires. Le problème $q-\mathsf{SDH}$ consiste, étant donné $(g, g^x, \ldots, g^{x^q}, \widetilde{g}, \widetilde{g}^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$, à générer une paire $(m, g^{\frac{1}{x+m}}) \in \mathbb{Z}_p \times \mathbb{G}_1$.

Ce problème, introduit par Boneh et Boyen [BB04], sous-tend la sécurité de leur schéma de signature que nous rappelons dans la section 3.2.2.

Définition 20. Soit $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires. Soient $\widetilde{X} = \widetilde{g}^x$ et $\widetilde{Y} = \widetilde{g}^y$ où x et y sont des scalaires aléatoires et $\widetilde{g}$ un générateur aléatoire de $\mathbb{G}_2$. On définit l'oracle $\mathcal{O}(m)$ qui, pour une entrée $m \in \mathbb{Z}_p$ retourne (h, h^y, h^{x+xym}) pour un un élément aléatoire h de $\mathbb{G}_1$. Le problème LRSW consiste, étant donnés $(\widetilde{X}, \widetilde{Y})$ ainsi qu'un accès illimité à $\mathcal{O}$, à générer un triplet similaire pour un message m^* qui n'a jamais été soumis à l'oracle.

Ce problème *interactif*, car il suppose l'accès à un oracle, a initialement été proposé par Lysyanskaya *et al.* [LRSW00]. La version décrite ici est adaptée aux groupes bilinéaires et permet de prouver la sécurité des signatures Camenisch-Lysyanskaya [CL04].

Définition 21. Soit $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires. Le problème $\mathsf{EXDH}-1$ consiste, étant donnés $(g, g^x, g^a, g^{y \cdot a}, g^z) \in \mathbb{G}_1^5$ et $(\widetilde{g}, \widetilde{g}^a, \widetilde{g}^y) \in \mathbb{G}_2^3$, à décider si $z = a \cdot x \cdot y$ ou si z est un scalaire aléatoire.

Définition 22. Soit $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires. Le problème $\mathsf{EXDH}-2$ consiste, étant donnés $(g, h, g^x, h^x, g^a, h^a, g^{y \cdot a}, h^{y \cdot a}, g^{z_1}, h^{z_2}) \in \mathbb{G}_1^{10}$ et $(\widetilde{g}, \widetilde{g}^a, \widetilde{g}^y) \in \mathbb{G}_2^3$, à décider si $z_1 = a \cdot x \cdot y = z_2$ ou si (z_1, z_2) est une paire aléatoire.

Définition 23. Soit $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires. Étant donnés $(g, g^a, g^x, \widetilde{g}, \widetilde{g}^a) \in \mathbb{G}_1^3 \times \mathbb{G}_2^2$, $(\{g^{y^i}\}_{i=1}^{i=n}, \{g^{x \cdot y^i}\}_{i=1}^{i=n-1}, \{\widetilde{g}^{1/y^i}\}_{i=1}^{i=n}) \in \mathbb{G}_1^{2n-1} \times \mathbb{G}_2^n$ et $g^z \in \mathbb{G}_1$, le problème $\mathsf{EMDDH}-1$ consiste à décider si $z = x \cdot y^n/a$ ou si z est un scalaire aléatoire.

Définition 24. Soit $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires. Étant donnés $(g, g^a, g^x, g^t, \widetilde{g}, \widetilde{g}^a) \in \mathbb{G}_1^4 \times \mathbb{G}_2^2$, $(\{g^{y^i}\}_{i=1}^{i=n}, \{g^{t \cdot y^i}\}_{i=1}^{i=n}, \{g^{x \cdot y^i}\}_{i=1}^{i=n-1}, \{g^{x \cdot t \cdot y^i}\}_{i=1}^{i=n-1}, \{\widetilde{g}^{1/y^i}\}_{i=1}^{i=n}) \in \mathbb{G}_1^{4n-2} \times \mathbb{G}_2^n$ et $(g^{z_1}, g^{z_2}) \in \mathbb{G}_1^2$, le problème $\mathsf{EMDDH}-2$ consiste à décider si $(z_1, z_2) = (x \cdot y^n/a, x \cdot t \cdot y^n/a)$ ou si (z_1, z_2) est une paire de scalaires aléatoires.

Le problème $\mathsf{EXDH}-1$ a été utilisé, sous un autre nom, dans [Duc10]. Les 3 autres problèmes ont été introduits dans nos articles *Divisible E-Cash Made Practical* [CPST15a] et *Scalable Divisible E-Cash* [CPST15b]. Ils permettent d'établir la sécurité des protocoles de monnaie électronique que nous décrivons dans le chapitre 4.

Remarque 3. Chaque problème présenté ci-dessus définit implicitement une hypothèse qui suppose le problème difficile. Par exemple, l'hypothèse DL suppose que le problème DL est difficile. Cela permettra de simplifier l'énoncé des théorèmes relatifs à la sécurité de nos schémas.

Remarque 4.

1. L'hypothèse $\mathsf{EXDH}-1$ implique l'hypothèse XDH dans $\mathbb{G}_1$, c'est-à-dire qu'un algorithme $\mathcal{A}$, capable de résoudre avec une probabilité non-négligeable le problème XDH dans $\mathbb{G}_1$, pourra être utilisé pour résoudre le problème $\mathsf{EXDH}-1$. En effet, les éléments $(g, g^x, g^{y \cdot a}, g^z)$ issus de ce dernier constituent une instance valide du problème XDH. La réponse retournée par $\mathcal{A}$ pourra donc servir à déterminer la nature de z.
2. Bien qu'a priori plus complexe que sa variante $\mathsf{EXDH}-1$, le problème $\mathsf{EXDH}-2$ lui est en fait équivalent. En effet, supposons l'existence d'un algorithme $\mathcal{A}$ résolvant ce dernier avec une probabilité non-négligeable. Il est alors possible de résoudre le problème $\mathsf{EXDH}-1$, comme le prouve le raisonnement suivant.

Soit $(g, g^x, g^a, g^{y \cdot a}, g^z) \in \mathbb{G}_1^5$ et $(\widetilde{g}, \widetilde{g}^a, \widetilde{g}^y) \in \mathbb{G}_2^3$ une instance $\mathsf{EXDH}-1$. On génère $h \leftarrow g^b$, pour un $b \in \mathbb{Z}_p$ connu, et on calcule h^x, h^a, $h^{y \cdot a}$ et h^z afin d'obtenir une instance $\mathsf{EXDH}-2$. Par hypothèse sur $\mathcal{A}$, la valeur suivante (appelée avantage $\mathtt{Adv}$) est non-négligeable.

$$\mathtt{Adv} = |\mathtt{Pr}[\mathcal{A}(., g^z, h^z) | z = x \cdot y \cdot a] - \mathtt{Pr}[\mathcal{A}(., g^{z_1}, h^{z_2})]|$$

Considérons maintenant les deux valeurs suivantes :

$$\mathtt{Adv}_1 = |\mathtt{Pr}[\mathcal{A}(., g^z, h^z) | z = x \cdot y \cdot a] - \mathtt{Pr}[\mathcal{A}(., g^z, h^z)]|$$
$$\mathtt{Adv}_2 = |\mathtt{Pr}[\mathcal{A}(., g^z, h^z)] - \mathtt{Pr}[\mathcal{A}(., g^{z_1}, h^{z_2})]|.$$

Comme $\mathtt{Adv}_1 + \mathtt{Adv}_2 \geq \mathtt{Adv}$, au moins l'un des deux avantages $\mathtt{Adv}_1$ ou $\mathtt{Adv}_2$ est non-négligeable. Si c'est le cas de $\mathtt{Adv}_1$, cela signifie que $\mathcal{A}$ permet de résoudre le problème $\mathsf{EXDH}-1$ et donc que l'équivalence est vérifiée. Sinon, cela signifie que $\mathcal{A}$ résout le problème XDH dans $\mathbb{G}_1$ (il suffit d'étendre l'instance (g, g^{z_1}, h, h^{z_2}) en générant les autres éléments avec des scalaires x, y et a connus) et donc également $\mathsf{EXDH}-1$, comme nous l'avons expliqué dans la première partie de cette remarque.
Malgré l'équivalence des problèmes $\mathsf{EXDH}-1$ et $\mathsf{EXDH}-2$, nous conservons les deux définitions afin de simplifier l'étude de la sécurité des protocoles décrits dans le chapitre 4.

Chapitre 3

Outils Cryptographiques

Sommaire

Nous présentons dans ce chapitre les pré-requis cryptographiques nécessaires à la compréhension des prochaines parties. Nous commencerons par la notion de *sécurité prouvée* qui est fondamentale pour la cryptographie moderne puis introduirons des primitives fréquemment utilisées par les protocoles décrits dans ce mémoire.

3.1 Méthodologie de la Sécurité Prouvée

Comme nous l'avons expliqué dans la section 2.2.3, la sécurité *parfaite* (au sens de la théorie de l'information) n'est pas réaliste dans la très grande majorité des cas. Cependant, les concessions faites par la sécurité calculatoire ne signifie pas pour autant qu'il faut abandonner toute approche rigoureuse dans l'évaluation de celle-ci. En particulier, la sécurité de toute nouvelle construction cryptographique doit être *prouvée*, c'est-à-dire qu'il est nécessaire de montrer qu'il est *difficile* de la *casser*. Pour cela, on utilise une technique appelée *réduction* que nous décrivons ci-dessous. Il reste cependant à définir précisément la sécurité attendue d'un schéma et ce que signifie *casser* celle-ci. Cela correspond à la conception d'un modèle de sécurité que nous décrivons dans la partie 3.1.2.

3.1.1 Preuve par Réduction

Reprenons l'exemple d'un schéma de chiffrement parfaitement sûr, tel que celui de Vernam. Sa sécurité correspond au fait qu'un chiffré n'apporte aucune information (pour une entité ne disposant pas de la clé) sur le message qu'il contient. Cela se traduit mathématiquement par

le fait que les distributions des chiffrés et des messages sont indépendantes. Cette définition mathématique est pratique car elle permet de prouver effectivement la sécurité du schéma.

L'approche de la sécurité calculatoire est différente. Elle commence par *supposer* qu'un certain problème, tel que ceux décrits dans la section 2.2.3, est difficile, puis prouve que la construction cryptographique est sûre sous cette hypothèse. Pour cela, elle décrit une *réduction* $\mathcal{R}$ montrant comment transformer n'importe quel algorithme (appelé adversaire) $\mathcal{A}$ réussissant à casser la sécurité de la construction avec une probabilité non négligeable en un algorithme réussissant à résoudre le problème supposé difficile. Le point important est que $\mathcal{R}$ ne fait aucune hypothèse sur $\mathcal{A}$, si ce n'est sur sa capacité à attaquer le schéma.

De cette manière, la sécurité de la construction cryptographique dépend directement de la difficulté du problème sous-jacent, censé être mieux étudié. C'est par exemple le cas du problème du logarithme discret (DL) qui a fait l'objet de très nombreux travaux et que l'on peut donc considérer comme difficile. Par conséquent, on pourra être confiant dans la sécurité d'un schéma prouvé sûr sous l'hypothèse DL car une attaque réussie contre celui-ci impliquerait la résolution de ce problème.

Cette approche a favorisé la formulation d'une multitude de nouvelles hypothèses offrant plus de possibilités que les problèmes classiques tels que celui du logarithme discret ou de la factorisation. Cependant, elle est également critiquée [KM10] par certains qui jugent que prouver la sécurité d'un schéma sous une nouvelle hypothèse mal connue n'apporte au final aucune garantie.

Néanmoins, les preuves par réduction, même sous de nouvelles hypothèses, restent souhaitables car elles permettent de ramener l'évaluation de la sécurité d'un protocole cryptographique, souvent complexe, à l'étude d'un problème mathématique clairement formulé. De plus, ces nouvelles hypothèses pourront alors être étudiées et utilisées par davantage de cryptologues, ce qui renforcera la confiance en leur véracité. C'est par exemple le cas de l'hypothèse Diffie-Hellman [DH76] ou de l'hypothèse RSA [RSA78] qui sont aujourd'hui considérées comme « classiques ».

3.1.2 Modèle de Sécurité

Nous avons, depuis le début de ce mémoire, fait à plusieurs reprises référence à la sécurité d'un schéma cryptographique. Cette notion, qui semble pourtant intuitive, est en fait particulièrement complexe à définir. Considérons par exemple un schéma de signature. Une première tentative pour en définir la sécurité pourrait être d'exiger qu'il soit difficile de contrefaire une signature. Bien que cette définition semble correcte, elle manque en fait de précision. Qu'entend-on exactement par « contrefaire une signature » ? Cela signifie-t-il qu'il doit être difficile de créer (on dit également *falsifier*) une signature sur un message qui n'a jamais été signé ? Est-ce que produire une nouvelle signature sur un message déjà signé peut être considéré comme une attaque valide ? Par ailleurs, un autre point à traiter concerne les capacités de l'adversaire. En effet, celui-ci n'a-t-il accès qu'à la clé publique de sa victime ? Peut-il accéder à des signatures valides ?

Cet exemple illustre bien le fait que la notion de « schéma sûr » n'a de sens qu'une fois qu'un *modèle de sécurité* précisant quels sont les objectifs et les capacités de l'adversaire est défini. En effet, un schéma sûr dans un certain modèle pourra ne plus l'être dans un autre. Nous verrons dans ce qui suit qu'il peut exister, pour une même primitive cryptographique (telle que la signature), plusieurs modèles de sécurité, chacun adapté à un contexte particulier.

Au final, la formalisation de la sécurité présente donc plusieurs avantages. Pour le cryptographe, elle permet de définir avec précision les propriétés de sécurité que son schéma doit atteindre mais également les attaques contre lesquelles il doit se prémunir. Pour l'utilisateur, elle permet de comparer efficacement les différentes constructions et de choisir la plus adaptée à ses besoins.

En pratique, un modèle de sécurité sera défini par des *jeux de sécurité* entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$, ce dernier cherchant à satisfaire une certaine condition de *gain* (par exemple, produire une signature valide sur un nouveau message). Pour cela il peut avoir à sa disposition

certains *oracles* représentant ses différentes capacités dans la réalité. Par exemple, la possibilité pour un adversaire d'obtenir des signatures sur un message de son choix sera modélisée par un oracle de signature que $\mathcal{A}$ pourra solliciter au cours du jeu de sécurité.

Le modèle de l'oracle aléatoire (ROM)

Comme nous l'avons vu avec la sécurité calculatoire, il est parfois nécessaire de faire certaines concessions sur la rigueur pour obtenir une meilleure efficacité. Une des possibilité peut être d'introduire un modèle idéalisé qui, bien qu'il ne reflète pas exactement la réalité, permet de se montrer raisonnablement confiant en la sécurité d'un schéma.

L'exemple le plus connu en cryptographie est celui du *modèle de l'oracle aléatoire* qui suppose l'existence d'une certaine fonction $\mathcal{H}$ que l'on ne peut évaluer que par des requêtes à un oracle. En pratique, $\mathcal{H}$ sera souvent une fonction de hachage cryptographique (voir section 3.4.1). Bien sûr, aucune implémentation de ces fonctions n'agit comme un oracle aléatoire mais on espère que leur fonctionnement se révèlera assez proche de ce dernier pour que le schéma reste sûr malgré tout.

Ce modèle, bien que critiqué [CGH98, CGH04], permet néanmoins d'obtenir des gages sérieux quant à la sécurité d'un schéma [BR93]. Depuis son introduction, il a permis la construction de nombreux protocoles très efficaces dont la sécurité n'a jamais été remise en cause jusque-là. Cependant, il est important de noter qu'une preuve dans le *modèle standard*, *i.e.* qui n'a recours à aucune idéalisation de ce type, reste toujours préférable.

3.2 Signature Numérique

L'article précurseur de Diffie et Hellman [DH76] n'a pas seulement proposé le concept de cryptographie à clé publique, il a également introduit une des primitives cryptographiques majeures : la signature numérique. Bien qu'initialement conçue pour fournir l'équivalent numérique d'une signature manuscrite, celle-ci est aujourd'hui fréquemment utilisée comme « brique cryptographique » servant à construire des protocoles plus complexes. Dans ce dernier cas, les propriétés exigées du schéma de signature vont au delà de la simple efficacité, celui-ci doit également être compatible avec les objectifs et les autres briques cryptographiques du protocole. Il existe une multitude de schémas de signature dont nous donnerons deux exemples dans la partie 3.2.2, mais nous commençons tout d'abord par définir cette primitive et présenter les modèles de sécurité associés.

3.2.1 Définition

Un schéma de signature numérique Σ est défini par les quatre algorithmes suivants :

- l'algorithme $\Sigma.\texttt{Setup}$ qui, prenant en entrée un paramètre de sécurité k, génère les paramètres publics *p.p.* du système ;
- l'algorithme de génération de clés $\Sigma.\texttt{Keygen}$ qui, prenant en entrée *p.p.*, génère un couple de clés $(\mathsf{sk}, \mathsf{pk})$ appelées respectivement *clé secrète* et *clé publique* (on suppose que *p.p.* $\subset \mathsf{pk}$ et que $\mathsf{pk} \subset \mathsf{sk}$) ;
- l'algorithme de signature $\Sigma.\texttt{Sign}$ qui, prenant en entrée la clé secrète sk et un message m, retourne une signature σ sur m ;
- et l'algorithme de vérification $\Sigma.\texttt{Verify}$ qui, prenant en entrée un message m, une signature σ et une clé publique pk, retourne 1 si σ est une signature valide sur m pour la clé pk et 0 sinon.

La propriété de sécurité attendue d'un schéma de signature est en général la *résistance aux falsifications existentielles contre des attaques adaptatives à messages choisis* [GMR88], notée EUF-CMA. Celle-ci signifie qu'il est difficile de produire une signature σ sur un message m,

$\mathtt{Exp}_{\mathcal{A}}^{\mathsf{EUF-CMA}}(1^k)$

1. $p.p. \leftarrow \Sigma\mathtt{Setup}(1^k)$
2. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma\mathtt{Keygen}(p.p.)$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sign}}(\mathsf{pk})$
4. Si m^* a été soumis à $\mathcal{O}\mathtt{Sign}$ alors retourner 0
5. Retourner $\mathtt{Verify}(m^*, \sigma^*, \mathsf{pk})$

Figure 3.1 – Sécurité EUF-CMA

même pour un adversaire pouvant obtenir des signatures sur un certain nombre de messages (différents de m, évidemment) de son choix. Elle est formellement définie par le jeu de sécurité suivant, noté $\mathtt{Exp}_{\mathcal{A}}^{\mathsf{EUF-CMA}}$, entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$.

- **Initialisation :** $\mathcal{C}$ exécute les algorithmes `Setup` et `Keygen` obtenant ainsi sk et pk. La clé publique est alors envoyée à pk.
- **Requêtes :** $\mathcal{A}$ soumet de manière adaptative (c'est-à-dire l'une après l'autre) des requêtes de signatures sur, au plus, q messages $m_1,\ldots,m_q$ auxquelles $\mathcal{C}$ répond en renvoyant $\sigma_i \leftarrow \mathtt{Sign}(\mathsf{sk}, m_i)$.
- **Résultat :** $\mathcal{A}$ produit finalement un couple message-signature (m^*, σ^*) et gagne le jeu si $\mathtt{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$ (*i.e.* la signature est valide) et si $m^* \neq m_i \ \forall i \in [1, q]$ (*i.e.* $\mathcal{A}$ n'a pas produit une signature sur un message déjà signé).

Un schéma de signature sera dit EUF-CMA sûr, si la probabilité de succès de tout adversaire probabiliste polynomial $\mathcal{A}$ peut être majorée par une fonction négligeable.

Dans la suite nous présenterons les jeux de sécurité d'une manière plus concise. Le résultat pour le jeu de sécurité que nous venons de décrire est donné dans la figure 3.1. La capacité de l'adversaire à demander des signatures sur des messages de son choix est désormais modélisée par un oracle $\mathcal{O}\mathtt{Sign}$ qui prend en entrée un message m et renvoie une signature dessus. La probabilité de succès d'un adversaire $\mathcal{A}$ peut alors être définie comme $\mathtt{Pr}(\mathtt{Exp}_{\mathcal{A}}^{\mathsf{EUF-CMA}}(1^k) = 1)$.

Bien que la sécurité EUF-CMA soit la plus classique, elle peut être trop forte, ou trop faible, dans certains cas. Nous verrons par exemple que le schéma de monnaie électronique que nous présentons au chapitre 4 nécessite une sécurité *forte*, c'est-à-dire qu'un adversaire ne doit pas non plus être capable de générer une nouvelle version d'une signature sur un message donné à l'oracle $\mathcal{O}\mathtt{Sign}$. Le jeu de la sécurité EUF-CMA *forte* [ADR02] (que nous noterons SUF-CMA) est alors défini de la même manière que dans la figure 3.1 sauf que la condition de l'étape 4 est changée en :

Si (m^*, σ^*) a été renvoyé par $\mathcal{O}\mathtt{Sign}$ alors retourner 0

Inversement, il est parfois possible de relâcher les exigences de sécurité afin de gagner en efficacité. Il existe par exemple la sécurité face aux attaques non-adaptatives à messages choisis (EUF-SCMA), où l'adversaire doit soumettre ses requêtes de signature avant d'avoir vu la clé publique, ou celle face aux attaques à message unique, où l'adversaire n'a le droit qu'à une seule requête de signature. Tous ces modèles correspondent à des situations concrètes que nous rencontrerons dans ce mémoire.

3.2.2 Quelques Exemples

L'émergence de la cryptographie à base de couplages a favorisé la conception de nouveaux schémas de signature compatibles avec cet environnement. Parmi eux, ceux proposés par Boneh et Boyen [BB04] et par Camenisch et Lysyanskaya [CL04] sont probablement les plus populaires.

Signatures Boneh-Boyen (BB)

Introduites en 2004, les signatures BB sont les premières signatures courtes à base de couplage, prouvées sûres dans le modèle standard (par opposition aux signatures proposées dans [BLS01] prouvées, elles, dans le ROM). L'article original décrit en fait deux schémas satisfaisant des propriétés de sécurité différentes. Nous choisissons ici de décrire le schéma le plus efficace satisfaisant la propriété de résistance aux falsifications existentielles contre des attaques sélectives à messages choisis (EUF-SCMA).

- $\texttt{Setup}(1^k)$: Soit k le paramètre de sécurité du système, cet algorithme retourne la description de groupes bilinéaires $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$.
- $\texttt{Keygen}(p.p.)$: L'algorithme sélectionne deux générateurs $g \in \mathbb{G}_1$ et $\widetilde{g} \in \mathbb{G}_2$ ainsi qu'un scalaire aléatoire $x \xleftarrow{\$} \mathbb{Z}_p$ et calcule $\widetilde{X} \leftarrow \widetilde{g}^x$ et $z \leftarrow e(g, \widetilde{g})$. La clé secrète sk est x et la clé publique pk est $(g, \widetilde{g}, \widetilde{X}, z)$.
- $\texttt{Sign}(\mathsf{sk}, m)$: Soit $m \in \mathbb{Z}_p$ le message à signer, l'algorithme retourne la signature $\sigma \leftarrow g^{\frac{1}{x+m}}$.
- $\texttt{Verify}(m, \sigma, \mathsf{pk})$: Pour vérifier la validité d'une signature σ sur un message m, cet algorithme teste si $e(\sigma, \widetilde{X} \cdot \widetilde{g}^m) = z$ auquel cas il retourne 1. Sinon, il retourne 0.

La propriété EUF-SCMA de ce schéma repose sur l'hypothèse $q - \mathsf{SDH}$ où q est le nombre maximal de requêtes de signatures que l'adversaire peut effectuer. La variante de ce schéma décrite dans le même article permet d'atteindre la propriété EUF-CMA mais au prix d'une légère augmentation de la complexité. Par ailleurs, comme expliqué dans [CL04] et détaillé dans [ASM06], il est possible de modifier ces algorithmes pour permettre la signature de plusieurs messages $(m_1, \ldots, m_r)$.

Signatures Camenisch-Lysyanskaya (CL)

Contrairement aux signatures BB qui visaient avant tout l'efficacité, les signatures CL ont principalement été conçues pour servir de brique de base à d'autres protocoles cryptographiques. Cela explique en grande partie leur popularité, celles-ci étant par exemple utilisées pour construire des schémas de signature de groupe [BCN+10], d'attestation anonyme [CPS10], de signature agrégée [LLY13] ou de monnaie électronique [CPST15a]. Nous décrivons ici la version générale permettant de signer r messages à la fois.

- $\texttt{Setup}(1^k)$: Soit k le paramètre de sécurité du système, cet algorithme retourne la description de groupes bilinéaires $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$.
- $\texttt{Keygen}(p.p.)$: L'algorithme sélectionne deux générateurs $g \in \mathbb{G}_1$ et $\widetilde{g} \in \mathbb{G}_2$ ainsi que des scalaires aléatoires $x, y, z_2, \ldots z_r \xleftarrow{\$} \mathbb{Z}_p$ et calcule $\widetilde{X} \leftarrow \widetilde{g}^x$, $\widetilde{Y} \leftarrow \widetilde{g}^y$, $\widetilde{Z}_2 \leftarrow \widetilde{g}^{z_1}, \ldots, \widetilde{Z}_r \leftarrow \widetilde{g}^{z_r}$. La clé secrète sk est $(x, y, z_1, \ldots, z_r)$ et la clé publique pk est $(g, \widetilde{g}, \widetilde{X}, \widetilde{Y}, \widetilde{Z}_2, \ldots, \widetilde{Z}_r)$.
- $\texttt{Sign}(\mathsf{sk}, (m_1, \ldots, m_r))$: Pour signer r messages, l'algorithme génère un élément $a \xleftarrow{\$} \mathbb{G}_1$ puis calcule :
  - $A_i \leftarrow a^{z_i}\ \forall\ 2 \leq i \leq r$
  - $b \leftarrow a^y$ et $B_i \leftarrow A_i^y\ \forall\ 2 \leq i \leq r$
  - $c = a^{x + xy \sum_{i=1}^{r} z_i m_i}$

  et retourne la signature $\sigma \leftarrow (a, b, c, \{(A_i, B_i)\}_{i=2}^r)$.
- $\texttt{Verify}(m, \sigma, \mathsf{pk})$: Pour tester la validité d'une signature $\sigma = (a, b, c, \{(A_i, B_i)\}_{i=2}^r)$ sur les messages $(m_1, \ldots m_r)$, cet algorithme vérifie chacune des égalités suivantes :
  - $e(a, \widetilde{Z}_i) = e(A_i, \widetilde{g})\ \forall\ 2 \leq i \leq r$
  - $e(a, \widetilde{Y}) = e(b, \widetilde{g})$ et $e(A_i, \widetilde{Y}) = e(B_i, \widetilde{g})\ \forall\ 2 \leq i \leq r$
  - $e(c, \widetilde{g}) = e(a \cdot b^{m_1} \prod_{i=2}^{r} B_i^{m_i}, \widetilde{X})$

  L'algorithme retourne 1 si toutes ces égalités sont vérifiées et 0 sinon.

La sécurité EUF-CMA de ces signatures a été prouvée sous l'hypothèse LRSW. À première vue, l'intérêt des signatures CL peut sembler limité comparé à celui des signatures BB qui offrent une meilleure efficacité et qui reposent sur une hypothèse jugée plus robuste. Cependant, il n'en est rien en raison de la flexibilité et des propriétés supplémentaires dont elles bénéficient. L'une des plus importantes d'entre elles est probablement la capacité de ces signatures à être régénérées, c'est-à-dire qu'étant donné une signature σ sur des messages $(m_1, \ldots, m_r)$, n'importe qui peut générer une nouvelle signature σ' différente de σ. En effet, si $\sigma = (a, b, c, \{(A_i, B_i)\}_{i=2}^r)$ est une signature valide sur $(m_1, \ldots, m_r)$ alors il en va de même pour $\sigma' = (a^t, b^t, c^t, \{(A_i^t, B_i^t)\}_{i=2}^r)$ quel que soit le scalaire $t \in \mathbb{Z}_p$ choisi. Le point important est qu'il est difficile, pour un adversaire ne connaissant pas $(m_1, \ldots, m_r)$, de faire le lien entre σ et σ' ce qui se révèle extrêmement utile pour l'anonymat de certains protocoles cryptographiques. En effet, leurs utilisateurs pourront à plusieurs reprises faire valoir leurs droits en révélant une signature délivrée par une autorité sans pouvoir être tracé (à condition de régénérer cette signature à chaque fois). Comme nous le verrons dans ce mémoire, cela permet de diminuer sensiblement la complexité des preuves à divulgation nulle que nous allons définir à présent.

3.3 Système de Preuves

En mathématiques, il est en général nécessaire de prouver toute affirmation. Prenons l'exemple d'une entité $\mathcal{P}$ (que nous appellerons *prouveur*) affirmant que le nombre 570741 n'est pas premier. Elle doit alors en fournir une preuve qui peut par exemple consister en la décomposition de ce nombre en facteurs non triviaux : $570741 = 969 \times 589$. En effet, n'importe quelle entité $\mathcal{V}$ (que nous appellerons *vérificateur*) est capable d'en vérifier la validité en effectuant une simple multiplication. Cependant, cette preuve n'a pas seulement révélé que 570741 n'était pas premier, elle a également fourni une information supplémentaire à savoir une factorisation possible.

Le problème de mesurer la quantité d'information révélée par une preuve a été étudié par Goldwasser, Micali et Rackoff [GMR89] qui proposèrent la notion de *preuve à divulgation nulle* devenue fondamentale pour la cryptographie moderne.

3.3.1 Preuve à Divulgation Nulle

Définition 25. Soit $\{0,1\}^*$ l'ensemble des chaînes finies de bits. Un *langage* $\mathcal{L}$ (respectivement une *relation* $\mathcal{R}$) est un sous-ensemble de $\{0,1\}^*$ (respectivement de $\{0,1\}^* \times \{0,1\}^*$). Toute relation $\mathcal{R}$ définit implicitement un langage $\mathcal{L}_\mathcal{R} = \{y \in \{0,1\}^* : \exists w \in \{0,1\}^* \text{ tel que } (y,w) \in \mathcal{R}\}$. L'élément w tel que $(y,w) \in \mathcal{R}$ est un *témoin* de y.

Exemple 2: Considérons par exemple la relation $\mathcal{R}$ définie sur $(\mathbb{N}^*)^2$ par $(y,w) \in \mathcal{R}$ si $w \notin \{1,y\}$ et si w divise y (noté $w|y$). Cette relation de divisibilité définit alors l'ensemble $\mathcal{L}_\mathcal{R} = \{y \in \mathbb{N}^* : \exists w \in \mathbb{N}^* \setminus \{1,y\} \text{ tel que } w|y\}$ qui correspond à l'ensemble des nombres strictement positifs non premiers. Par conséquent $570741 \in \mathcal{L}_\mathcal{R}$ car il est divisible par 969 qui en est donc un témoin. On peut d'ailleurs remarquer qu'il n'y a pas unicité du témoin car 589 en est également un.

Définition 26. Un système de preuve pour une relation $\mathcal{R}$ est un protocole interactif entre un prouveur $\mathcal{P}$ et un vérificateur $\mathcal{V}$ tel que :

- $\forall (y,w) \in \mathcal{R}$, $\mathcal{P}$ prend en entrée (y,w) et $\mathcal{V}$ prend en entrée y. A la fin de l'interaction, $\mathcal{V}$ retourne un bit $b \in \{0,1\}$.
- *Complétude :* $\forall (y,w) \in \mathcal{R}$, $\mathcal{V}$ retourne 0 avec probabilité négligeable.
- *Validité :* $\forall y \in \{0,1\}^*$, si $\mathcal{V}$ retourne 1 avec probabilité non négligeable alors il existe w tel que $(y,w) \in \mathcal{R}$.

Le vérificateur $\mathcal{V}$ retourne 1 s'il accepte la preuve et 0 sinon. Ce protocole permet donc à $\mathcal{P}$ de convaincre $\mathcal{V}$ de l'appartenance de y au langage $\mathcal{L}_\mathcal{R}$. La propriété de validité assure que même

un prouveur malhonnête ne peut convaincre à tort $\mathcal{V}$ de l'appartenance d'un élément à ce même ensemble avec une probabilité non négligeable.

La question posée par Goldwasser Micali et Rackoff est de savoir comment mesurer l'information révélée par ce type de protocoles et, plus particulièrement, s'il est possible de concevoir un protocole, dit *à divulgation nulle*, ne révélant aucune autre information que l'appartenance de y au langage $\mathcal{L}_{\mathcal{R}}$.

Définition 27. Un système de preuve pour une relation $\mathcal{R}$ est dit *à divulgation nulle* si, pour tout vérificateur $\mathcal{V}$ s'exécutant en temps polynomial, il existe un simulateur $\mathcal{S}$, tel que $\forall(y, w) \in \mathcal{R}$, $\mathcal{S}(y)$ retourne une chaîne $\texttt{str} \in \{0,1\}^*$ indistinguable des traces des communications entre $\mathcal{P}(y, w)$ et $\mathcal{V}(y)$.

Pour formaliser la notion de *divulgation nulle*, cette définition considère les traces des communications (*i.e.* les éléments échangés) entre un prouveur $\mathcal{P}$ connaissant le couple (y, w) et n'importe quel vérificateur $\mathcal{V}$ connaissant y. Si ces traces peuvent être *simulées sans la connaissance d'un témoin* w cela signifie que la preuve n'apporte aucune information sur celui-ci.

Bien que ne connaissant pas w, $\mathcal{S}$ dispose d'un avantage sur $\mathcal{P}$. En effet, il peut faire plusieurs essais avant d'obtenir une simulation valide alors que le prouveur sera rejeté en cas d'échec. En général, les simulateurs utilisent la technique du *rembobinage* permettant de revenir en arrière dans le protocole en fonction des réponses de $\mathcal{V}$.

La propriété de *divulgation nulle* est très forte, ce qui peut rendre difficile la conception de protocoles la satisfaisant. Il est cependant possible, dans certains cas, de reposer sur une notion plus faible appelée *indistinguabilité des témoins*.

Définition 28. Un système de preuve pour une relation $\mathcal{R}$ satisfait la propriété d'indistinguabilité des témoins si, pour tout vérificateur $\mathcal{V}$ s'exécutant en temps polynomial, tout $y \in \{0,1\}^*$ et tout couple (w_1, w_2) tel que $(y, w_1) \in \mathcal{R}$ et $(y, w_2) \in \mathcal{R}$ les traces des communications entre $\mathcal{P}(y, w_1)$ et $\mathcal{V}(y)$ sont indistinguables de celles entre $\mathcal{P}(y, w_2)$ et $\mathcal{V}(y)$.

Cette propriété signifie donc qu'aucun adversaire ne peut déterminer quel témoin a été utilisé par $\mathcal{P}$ durant la preuve. Si nous reprenons l'exemple précédent, cela signifie qu'il n'est pas possible de savoir si $\mathcal{P}$ a utilisé 969 ou 589 pour prouver que 570741 était factorisable.

3.3.2 Preuve de Connaissance

Comme l'ont fait remarquer Feige, Fiat et Shamir [FFS87], le terme de *divulgation nulle* est en fait un peu trompeur. En effet, lorsque je prouve avec ce type de preuve qu'un élément y appartient à un langage $\mathcal{L}$ je révèle de fait de l'information, à savoir que $y \in \mathcal{L}$. Ils proposèrent donc le concept de *preuve de connaissance*, par opposition aux preuves précédentes dite d'*appartenance*, permettant à $\mathcal{P}$ de prouver qu'il connait quelque chose, par exemple la situation de y vis-à-vis du langage $\mathcal{L}$.

En pratique, ces protocoles sont surtout utilisés pour prouver la connaissance d'un témoin w d'un élément $y \in \mathcal{L}$. En effet, considérons l'exemple d'un groupe cyclique $\mathbb{G}$ de générateur g. Il est possible de définir le langage $\mathcal{L} = \{y \in \{0,1\}^* : \exists w \in \mathbb{N} \text{ tel que } y = g^w\}$ qui est constitué de tous les éléments du groupe $\mathbb{G}$. Une preuve d'appartenance au langage $\mathcal{L}$ n'a ici pas vraiment de sens car il suffit simplement de s'assurer que $y \in \mathbb{G}$. En revanche prouver la connaissance du témoin w (qui est le logarithme discret de y en base g) peut être intéressant car il ne s'agit pas, pour le coup, d'une information triviale. Nous verrons d'ailleurs que lorsque y est la clé publique d'un utilisateur et w sa clé secrète, prouver la connaissance de cette dernière peut lui permettre de s'authentifier.

Comme souvent, la formalisation de cette notion intuitive est particulièrement complexe et a fait l'objet de plusieurs travaux [FFS87, TW87, BG93]. Nous choisissons ici d'en donner une définition un peu moins formelle et renvoyons vers la dernière référence pour plus de détails.

Définition 29. Un système de preuve à divulgation nulle pour une relation $\mathcal{R}$ est un *système de preuve de connaissance* si, pour tout prouveur $\mathcal{P}$ accepté avec probabilité non négligeable par un vérificateur $\mathcal{V}$ pour $y \in \{0,1\}^*$ et w tel que $(y,w) \in \mathcal{R}$, il existe un *extracteur* $\mathcal{E}(y)$ capable, en contrôlant l'exécution de $\mathcal{P}(y,w)$, de retourner un témoin w' valide pour y.

L'extracteur $\mathcal{E}$ ne connait que y mais peut interagir avec $\mathcal{P}$. Par conséquent, si $\mathcal{E}$ est capable de retrouver un témoin de y c'est nécessairement grâce à $\mathcal{P}$ qui devait donc le *connaître*.

Remarque 5. Tel que défini, l'existence d'un extracteur implique nécessairement la propriété de validité du système de preuve. En effet, si $\mathcal{V}$ accepte avec probabilité non négligeable alors il est possible d'extraire un témoin de y, ce qui prouve bien que $y \in \mathcal{L}_R$. Pour prouver qu'un protocole définit une preuve de connaissance il ne sera donc plus nécessaire d'en prouver la validité, exhiber un extracteur suffira.

Là encore, les propriétés attendues d'un système de preuve de connaissance peuvent sembler très fortes. Cependant, il a été prouvé [FFS87] qu'il est possible de construire de tels systèmes pour une catégorie très large de langages $\mathcal{L}$. Malheureusement, ce résultat reste théorique et ne décrit pas de constructions utilisables en pratique. Il existe néanmoins plusieurs protocoles efficaces adaptés à certains cas précis, l'un des plus célèbres étant celui dû à Schnorr [Sch90] permettant à un prouveur $\mathcal{P}$ de prouver la connaissance d'un logarithme discret dans un groupe d'ordre premier.

Protocole de Schnorr

Soient g et h deux éléments d'un groupe $\mathbb{G}$ d'ordre premier p. Le protocole de Schnorr, décrit dans la Figure 3.2, permet à $\mathcal{P}$ de prouver connaissance de w tel que $h = g^w$. On utilisera dans la suite la notion usuelle $PK\{(w) : h = g^w\}$ où les éléments entre parenthèses correspondent aux secrets du prouveur.

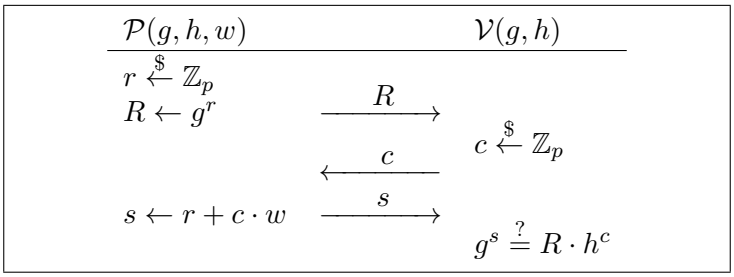


Figure 3.2 – Protocole de Schnorr

Ce type de protocoles en 3 passes, appelé Σ-protocole, est très utile en cryptographie, notamment parce qu'il peut être rendu *non-interactif* (*i.e.* en une passe) comme nous le verrons dans la partie suivante. La première phase du protocole, jusqu'à l'envoi de R est appelée *mise en gage*, la deuxième, correspondant à la sélection et à l'envoi de c est appelée *challenge* et la dernière est appelée *réponse*.

La complétude de ce protocole est simple à vérifier : si $h = g^w$ alors $g^s = g^r \cdot (g^w)^c = R \cdot h^c$. La construction de l'extracteur repose sur le fait que si $\mathcal{P}$ est capable, pour une même mise en gage de R, de répondre à deux challenges différents c et c' alors il est possible de retrouver w. En effet, soient s et s' les réponses correspondantes et $x = (s - s') \cdot (c - c')^{-1} \bmod p$, alors :

$$g^x = (g^s \cdot g^{-s'})^{(c-c')^{-1}} = (R \cdot h^c \cdot R^{-1} \cdot h^{-c'})^{(c-c')^{-1}} = h$$

et donc $x = w$. La propriété de divulgation nulle de ce protocole est en revanche plus problématique. En effet, si le vérificateur suit le protocole (on dit alors qu'il est *honnête*), alors les traces d'une

exécution du protocole sont $(R, c, s) \in \mathbb{G} \times \mathbb{Z}_p^2$ où c est choisi aléatoirement et *indépendamment de R*. Dans ce cas il est possible de construire un simulateur $\mathcal{S}$ qui :

1. sélectionne $c, s \xleftarrow{\$} \mathbb{Z}_p$;
2. calcule $R \leftarrow g^s \cdot h^{-c}$;
3. retourne (R, c, s).

Le triplet (R, c, s) ainsi construit possède la même distribution que les traces de l'exécution du protocole entre $\mathcal{P}$ et un vérificateur $\mathcal{V}$ *honnête*. Mais qu'en est-il si $\mathcal{V}$ ne suit pas exactement le protocole (ce qui est possible d'après la définition de preuve à divulgation nulle) ? Par exemple, $\mathcal{V}$ pourrait choisir son challenge c en fonction de R et pourrait donc distinguer un triplet construit par $\mathcal{S}$. Ce simulateur ne permet donc pas de montrer que ce protocole est à divulgation nulle. En revanche, il permet de prouver que ce dernier vérifie une propriété plus faible, à savoir qu'il est à divulgation nulle *face à un vérificateur honnête*.

Cette dernière propriété peut surprendre. En effet, elle sous-entend que le prouveur n'est protégé que face à un vérificateur *honnête*, ce qui ne semble pas très utile. Nous allons cependant voir qu'elle peut être suffisante pour construire des protocoles non-interactifs à divulgation nulle.

3.3.3 Preuve Non-Interactive

Le protocole de Schnorr décrit ci-dessus permet de prouver la connaissance d'un logarithme discret mais nécessite d'interagir avec le vérificateur $\mathcal{V}$. Dans certains cas, cela peut poser problème. Considérons par exemple le cas d'un paiement électronique. L'acheteur $\mathcal{P}$ peut être amené à prouver au marchand $\mathcal{V}$ que son paiement est valide, ce qui est possible avec une preuve interactive. Cependant, lorsque le commerçant souhaitera obtenir une compensation (c'est-à-dire l'équivalent en argent « réel » du montant de la transaction) auprès de sa banque, il lui sera impossible de prouver la validité du paiement. Les traces du protocole de preuve exécuté entre $\mathcal{P}$ et $\mathcal{V}$ n'ont en effet aucune valeur aux yeux d'une tierce partie car elles peuvent être simulées. Les *preuves non-interactives*, qui consistent en un unique message envoyé par le prouveur, apportent une solution à ce problème.

Cependant, effectuer une preuve en une seule passe n'est pas simple et nécessite en pratique de reposer sur une *chaîne de référence commune*, notée *CRS*, connue de $\mathcal{P}$ et $\mathcal{V}$ et générée par une entité de confiance.

La méthode Fiat-Shamir

En 1986, Fiat et Shamir [FS87] proposèrent une méthode générique permettant de convertir les Σ-protocoles à divulgation nulle face à un vérificateur honnête, tels que celui de Schnorr, en protocoles non-interactifs à divulgation nulle. Cette méthode est depuis couramment utilisée en cryptographie en raison de sa très grande efficacité mais également car les preuves non-interactives qu'elle génère peuvent être transformées, sans surcoût, en schémas de signature numérique (voir également [PS00]).

Soit $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$ une fonction pseudo-aléatoire (en pratique, on utilisera une fonction de hachage cryptographique définie dans la prochaine section). La méthode de Fiat-Shamir appliquée au protocole de Schnorr consiste à calculer le challenge c comme $\mathcal{H}(th||R)$, où th est une description des éléments impliqués dans la preuve et R est la mise en gage, plutôt qu'à le générer aléatoirement. Plus précisément, pour prouver la connaissance de w tel que $h = g^w$, $\mathcal{P}$ calcule :

1. $R \leftarrow g^r$ pour un certain $r \xleftarrow{\$} \mathbb{Z}_p$
2. $c \leftarrow \mathcal{H}(g||h||R)$
3. $s \leftarrow r + c \cdot w$

et envoie la preuve (R, c, s) à $\mathcal{V}$ qui la vérifie en testant si $g^s \stackrel{?}{=} R \cdot h^c$ et si $c \stackrel{?}{=} \mathcal{H}(g||h||R)$.

Pour construire un simulateur $\mathcal{S}$ et un extracteur $\mathcal{E}$, il va cependant être nécessaire de modéliser $\mathcal{H}$ comme une fonction ne pouvant être évaluée qu'au travers de requêtes à un oracle : c'est le modèle de l'oracle aléatoire (ROM) défini dans la partie 3.1.2. Pour simuler une preuve, $\mathcal{S}$ va sélectionner deux éléments aléatoires $c, s \xleftarrow{\$} \mathbb{Z}_p$, calculer $R \leftarrow g^s \cdot h^{-c}$ et fixer (car elle contrôle $\mathcal{H}$) $\mathcal{H}(g||h||R) = c$, c'est-à-dire qu'à toute requête à l'oracle pour évaluer $\mathcal{H}(g||h||R)$, $\mathcal{S}$ répondra c. L'extracteur $\mathcal{E}$ va de son côté, après une exécution du protocole sur une mise en gage R, rembobiner $\mathcal{P}$ pour changer la valeur $\mathcal{H}(g||h||R) = c$ en $c' \neq c$ et donc forcer $\mathcal{P}$ à répondre à 2 challenges différents pour une même mise en gage, ce qui permet d'extraire $w \leftarrow (s-s') \cdot (c-c')^{-1}$.

Le protocole ainsi construit est donc bien une preuve de connaissance non-interactive, dans le ROM, du logarithme discret de h en base g. Il est par ailleurs possible de convertir ce protocole en schéma de signature. En effet, si $\mathcal{P}$ calcule lors de la deuxième étape $c \leftarrow \mathcal{H}(g||h||R||m)$ alors (R, c, s) devient une signature sur le message m que $\mathcal{V}$ peut vérifier de la même manière que pour la preuve. Les schémas de signature ainsi construit sont souvent appelées *signature de connaissance*, notée SK.

Comme tous les protocoles reposant sur le ROM, la méthode Fiat-Shamir peut être critiquée en raison de son recours à ce modèle « idéal ». Cependant, elle a longtemps été le seul moyen efficace d'obtenir des preuves non interactives, les autres techniques génériques souffrant d'une complexité prohibitive. Ce n'est qu'en 2008 que Groth et Sahai [GS08] proposèrent le premier système efficace de preuves non interactives, prouvé sûr dans le modèle standard. Leur solution, adaptée aux environnements bilinéaires, a depuis fait l'objet de multiples applications permettant notamment de construire les premiers schémas de signature de groupe [Gro07] ou de monnaie électronique [IL13, CPST15a] dans le modèle standard.

Système de Preuves Groth-Sahai

Les preuves Groth-Sahai reposent sur l'existence d'une chaine de référence commune (CRS) qui peut être générée de manière à obtenir soit des preuves vérifiant la propriété de validité, soit des preuves vérifiant celle d'indistingabilité des témoins. Le point important est qu'il est difficile, sous une hypothèse dépendant des groupes bilinéaires utilisés, de déterminer quel type de CRS a été généré. Par exemple, dans le cas de l'utilisation de groupes bilinéaires de type 3, la difficulté de distinguer les deux types de CRS repose sur l'hypothèse SXDH définie dans la section 2.2.3.

Il n'est pas nécessaire, pour une bonne compréhension du reste de ce mémoire, de savoir comment fonctionne ces preuves mais simplement de connaître le type de relations qu'elles permettent de prouver. Nous en donnerons donc ici l'idée générale et renvoyons vers l'article [GS08] pour plus de détails.

Pour prouver que certains éléments, appelés variables, vérifient un ensemble de relations dans des groupes bilinéaires $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, le prouveur $\mathcal{P}$ les *met en gage* (voir section 3.4.2 ci-dessous) et calcule ensuite une preuve par relation. Ces relations peuvent être soit des équations de produits de couplages, soit des équations d'exponentiations multiples.

– Les premières sont du type :

$$\prod_{i=1}^{n} e(A_i, \widetilde{X}_i) \prod_{i=1}^{n} e(X_i, \widetilde{B}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_i, \widetilde{X}_j)^{a_{i,j}} = t_T$$

pour des variables $\{X_i\}_{i=1}^n \in \mathbb{G}_1$, $\{\widetilde{X}_i\}_{i=1}^n \in \mathbb{G}_2$ et des éléments publics, appelés *constantes*, $t_T \in \mathbb{G}_T$, $\{A_i\}_{i=1}^n \in \mathbb{G}_1$, $\{\widetilde{B}_i\}_{i=1}^n \in \mathbb{G}_2$ et $\{a_{i,j}\}_{i,j=1}^n \in \mathbb{Z}_p$.

– Les deuxièmes sont de la forme :

$$\prod_{i=1}^{n} A_i^{y_i} \prod_{j=1}^{n} X_j^{b_j} \prod_{i=1}^{n} \prod_{j=1}^{n} X_j^{y_i \cdot a_{i,j}} = T$$

pour des variables $\{X_i\}_{i=1}^n \in \mathbb{G}_k$, $\{y_i\}_{i=1}^n \in \mathbb{Z}_p$ et des constantes $T \in \mathbb{G}_k$, $\{A_i\}_{i=1}^n \in \mathbb{G}_k$, $\{b_i\}_{i=1}^n \in \mathbb{Z}_p$, $\{a_{i,j}\}_{i,j=1}^n \in \mathbb{Z}_p$ avec $k \in \{1, 2\}$.

Groth et Sahai ont également montré qu'il est possible de rendre les preuves d'équations d'exponentiations multiples à divulgation nulle. Il est possible de faire de même pour certaines équations de produits couplages mais nous n'en aurons pas besoin dans ce mémoire.

Remarque 6. Il est important de noter que les preuves Groth-Sahai ne sont pas nécessairement des preuves de connaissance. En effet s'il est possible d'extraire simplement les éléments $X \in \mathbb{G}_1$ et $\widetilde{X} \in \mathbb{G}_2$ des mises en gage, il n'en va pas de même des scalaires $y_i \in \mathbb{Z}_p$. Ce type de preuves n'est donc généralement pas utilisé pour prouver la connaissance d'un logarithme discret.

3.4 Autres Primitives Cryptographiques

Outre les signatures numériques et les preuves de connaissance, que nous utiliserons très fréquemment dans ce mémoire, nous ferons également appel à deux autres primitives cryptographiques, la fonction de hachage et la mise en gage.

3.4.1 Fonction de Hachage Cryptographique

Une fonction de hachage $\mathcal{H}$ est une fonction transformant n'importe quelle chaîne de bits en une autre de taille fixe, appelée *empreinte*. Elle est dite *cryptographique* si elle satisfait les deux conditions suivantes.

- *Résistance à la pré-image :* pour tout y appartenant à l'image de $\mathcal{H}$, il est difficile de trouver $x \in \{0,1\}^*$ tel que $\mathcal{H}(x) = y$.
- *Résistance aux collisions :* il est difficile de trouver x et x' appartenant à $\{0,1\}^*$ tels que $\mathcal{H}(x) = \mathcal{H}(x')$.

Une fonction vérifiant la première de ces propriétés est également dite *à sens unique*. Les fonctions de hachage cryptographiques font l'objet de standards tels que SHA-256 [SHA02] ou SHA-3 [SHA14] dont l'utilisation est aujourd'hui recommandée.

3.4.2 Mise en Gage

Un schéma de *mise en gage* cryptographique permet de masquer une donnée tout en gardant la possibilité de la révéler ultérieurement. Il peut être comparé à une boîte verrouillée dans laquelle serait enfermé un objet. Celui-ci resterait secret jusqu'à ce que le détenteur de la clé choisisse de le révéler en ouvrant la boîte.

De manière plus formelle, un schéma de mise en gage est défini par les 3 algorithmes suivants :

- l'algorithme `Setup` qui, prenant en entrée un paramètre de sécurité k, génère les paramètres publics *p.p.* du système ;
- l'algorithme `Commit` qui, prenant en entrée *p.p.* et un message m, retourne une mise en gage C ainsi qu'une valeur d'ouverture r ;
- l'algorithme `Open` qui, prenant en entrée C, r et m, retourne 1 si C est une mise en gage valide du message m pour l'ouverture r et 0 sinon.

Il est également possible de définir un schéma de *mise en gage extractible* en imposant à l'algorithme `Setup` de retourner, en plus de *p.p.*, une clé d'extraction `ek` et en rajoutant la description d'un algorithme `Extract` qui, prenant en entrée `ek` et une mise en gage C d'un message m, retourne m.

Sécurité. Pour reproduire les fonctionnalités d'une boîte verrouillée, un schéma de mise en gage doit être *contraignant* et *confidentiel*. Ces deux propriétés sont définies ci-dessous.

- Un schéma de mise en gage est calculatoirement contraignant si aucun adversaire probabiliste polynomial n'est capable de produire une mise en gage C ainsi que 2 paires différentes (r, m) et (r', m') telles que $\texttt{Open}(C, r, m) = 1 = \texttt{Open}(C, r', m')$.
- Un schéma de mise en gage est calculatoirement confidentiel si aucun adversaire probabiliste polynomial n'est capable de distinguer, parmi deux messages de son choix, lequel est contenu dans une mise en gage C.

Exemple 3: Pedersen [Ped92] a proposé le schéma de mise en gage suivant, défini sur un groupe $\mathbb{G}$ d'ordre premier p.

- $\texttt{Setup}(1^k)$: soit k le paramètre de sécurité, cet algorithme retourne $p.p. \leftarrow (g, h)$, où g et h sont deux générateurs de $\mathbb{G}$.
- $\texttt{Commit}(m)$: pour mettre en gage un message $m \in \mathbb{Z}_p$, cet algorithme génère un scalaire r aléatoire et retourne $(C, r) \leftarrow (g^r \cdot h^m, r)$.
- $\texttt{Open}(C, r, m)$: cet algorithme teste si $C = g^r \cdot h^m$, auquel cas il retourne 1. Sinon, il retourne 0.

Ce protocole peut être prouvé calculatoirement contraignant sous l'hypothèse du logarithme discret. On peut également noter qu'il vérifie une propriété plus forte que la confidentialité calculatoire car le message m est parfaitement masqué (au sens de la théorie de l'information).

Deuxième partie

Mécanismes Cryptographiques pour la Protection de la Vie Privée

Chapitre 4

Techniques Cryptographiques pour le Paiement Anonyme

Sommaire

Nous présentons dans ce chapitre des outils cryptographiques pour le paiement anonyme. Nous commençons par introduire le concept de *monnaie électronique* avant d'en présenter une variante appelée *monnaie électronique divisible*. Nous décrivons ensuite deux constructions respectant les contraintes d'efficacité d'un système de paiement et satisfaisant les exigences d'anonymat les plus strictes. Celles-ci sont issues des articles *Divisible E-Cash Made Practical* [CPST15a], publié à la conférence PKC 2015, et *Scalable Divisible E-Cash* [CPST15b], publié à la conférence ACNS 2015.

4.1 Monnaie Électronique

Les moyens de paiement électroniques offrent aujourd'hui un confort d'utilisation incomparable à celui de la monnaie traditionnelle. En effet, leurs utilisateurs se retrouvent débarrassés des problèmes d'appoint et de rendue de monnaie tout en bénéficiant d'une vitesse de transaction de l'ordre de la seconde. Malheureusement, chacun de leurs achats est aujourd'hui remonté à la banque, ce qui est bien loin d'être anodin. C'est même suffisant pour connaitre leurs goûts, leurs activités, leur croyances ou même leur état de santé.

Pourtant, les moyens de paiement électroniques ne sont pas nécessairement incompatibles avec la protection de la vie privée, comme le prouva Chaum lorsqu'il introduisit en 1982 le concept de *monnaie électronique* anonyme [Cha82].

Une monnaie électronique vise à reproduire, de manière numérique, le fonctionnement des espèces traditionnelles. En pratique, ces systèmes considèrent trois types d'entités : la *banque*, qui émet la monnaie, les *utilisateurs*, qui retirent des pièces auprès d'elle, et les *marchands*, chez qui les utilisateurs les dépensent. Dans l'idéal, l'ensemble des utilisateurs et celui des marchands devraient être confondus, c'est-à-dire qu'une personne recevant une pièce (donc un marchand) devrait à son tour pouvoir la dépenser (devenant ainsi un utilisateur). Malheureusement, un tel système, qualifié de *transférable*, souffre d'une forte complexité qui lui est inhérente [CP93]. Pour des raisons d'efficacité, on utilisera donc en pratique un système plus simple où les marchands doivent déposer à la banque les pièces qu'ils ont reçues, ne pouvant donc pas les dépenser à nouveau.

La monnaie électronique présente cependant une différence fondamentale avec la monnaie traditionnelle : elle est aisément reproductible. Cette propriété, propre à toute donnée numérique, pose un véritable problème : comment la banque peut-elle empêcher un utilisateur malhonnête de réutiliser plusieurs fois une même pièce ? Si le système n'était pas anonyme, il lui suffirait d'additionner l'ensemble des retraits d'un utilisateur ainsi que ses dépenses et de s'assurer que la différence reste positive. Malheureusement, cela est totalement impossible pour un système de monnaie électronique pour lequel on souhaite justement éviter cette traçabilité. Toute la difficulté de la conception de ce type de système est donc de fournir à la banque un moyen de détecter les fraudes (appelées aussi double-dépenses) sans remettre en cause l'anonymat.

Pour résoudre ce problème, Chaum proposa le concept de *signature aveugle*. Cette primitive, formalisée dans [PS96], permet à un utilisateur d'obtenir une signature σ sur un message m inconnu du signataire. Par ailleurs, ce dernier est incapable, lorsqu'il voit passer la paire (σ, m), de savoir à quel moment il a émis cette signature. Dans le contexte de la monnaie électronique, une pièce va consister en une signature aveugle reçue de la banque par l'utilisateur. Ce dernier pourra ainsi la présenter à un marchand qui en vérifiera la validité à l'aide de la clé publique du système. Chaque fois qu'un marchand déposera une pièce à la banque, celle-ci conservera dans un registre la signature σ associée. Cela lui permettra de détecter les double-dépenses (car une même signature apparaîtra alors pour plusieurs dépôts) mais pas d'identifier les utilisateurs car elle sera incapable, en raison des propriétés des signatures aveugles, de savoir à qui elle a délivré la signature σ.

En théorie, le problème du paiement électronique a donc été résolu par la construction de schémas de signatures aveugles efficaces (voir [PS96] pour quelques exemples). Malheureusement, l'usage de la monnaie électronique ainsi construite reste problématique. En effet, comment être capable de gérer n'importe quel montant ?

Une première solution, directement inspirée des espèces traditionnelles, serait de distribuer des pièces électroniques de montants différents. En pratique, cela se traduirait par une banque qui utiliserait plusieurs paires $\{(\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_r, \mathsf{pk}_r)\}$ de clés de signatures aveugles, chacune étant associée à une certaine valeur. Cependant, supposons qu'un utilisateur dispose d'une pièce d'un montant de 10 € et souhaite en dépenser 8 €. Il devrait alors soit obtenir de la monnaie auprès de sa banque pour pouvoir faire l'appoint, soit se faire rendre la monnaie par le commerçant. La première possibilité n'est pas souhaitable en terme d'usage car elle nécessite d'être connecté lors de la transaction. La deuxième ne l'est pas non plus car l'utilisateur deviendrait alors un « marchand » (car il recevrait une pièce) et perdrait du coup son anonymat (seuls les utilisateurs sont anonymes).

Une autre solution serait de n'utiliser que des pièces du plus petit montant possible, à savoir 0.01 €. Cependant, cela signifierait que l'utilisateur devrait stocker plusieurs milliers de signatures et, lors d'une dépense, en envoyer plusieurs centaines au marchand qui devrait toutes les vérifier. Dans leur article *Compact E-Cash* [CHL05], Camenisch, Hohenberger et Lysyanskaya ont partiellement résolu ce problème en proposant le premier schéma capable de stocker très efficacement plusieurs pièces à la fois. Malheureusement, celles-ci doivent toujours être dépensées une par une, ce qui reste inenvisageable en pratique.

La dernière solution à notre problème a été proposée par Okamoto et Ohta [OO92] sous le nom de *monnaie électronique divisible*. Cette variante de la monnaie électronique permet à un utilisateur de retirer une pièce d'un certain montant V puis de la dépenser en plusieurs fois. Plus précisément, cette pièce permet d'effectuer différentes transactions de montants $v_1, \ldots, v_r$ jusqu'à avoir $\sum_i v_i = V$. Cette solution semble donc la plus adaptée à un usage réel et a fait, depuis son introduction, l'objet de nombreux travaux (par exemple [Oka95, NS00, CG07, CG10, IL13]).

4.2 Monnaie Électronique Divisible

La plupart des schémas de monnaie électronique divisible suivent la même idée. Chaque pièce est associée à un arbre binaire contenant 2^n feuilles disposant toutes d'un numéro de série. Cet arbre est construit de sorte que la connaissance d'informations relatives à l'un de ses sommets s permet de retrouver tous les numéros de série des feuilles qui en descendent. Ainsi, lorsqu'un utilisateur souhaite dépenser un montant de 2^l, il va révéler les informations associées à un sommet s de profondeur $n-l$ (c'est-à-dire qui est l'ancêtre de 2^l feuilles) qui n'a jamais été dépensé jusque-là. Ces informations vont permettre à la banque de retrouver, puis de conserver, les 2^l numéros de série associés. L'utilisateur honnête sera donc capable, à chaque fois, de choisir un sommet non dépensé et de révéler ainsi de nouveaux numéros de série. En revanche, un fraudeur sera lui obligé, une fois qu'il aura dépensé l'intégralité de sa pièce, d'utiliser un sommet déjà dépensé, associé à des numéros de série déjà connus de la banque. Celle-ci sera donc capable de détecter très simplement la double-dépense.

Là encore, la difficulté majeure est de concevoir un arbre qui bénéficie de ces propriétés sans mettre en danger l'anonymat de l'utilisateur. À ce titre, les premières constructions [OO92, Oka95, CFT98] n'offraient qu'un niveau d'anonymat assez faible, chaque utilisation d'une même pièce pouvant être tracée. Cela était dû en pratique à l'utilisation de signatures aveugles qui devaient être montrées à chaque utilisation de la pièce. Par la suite, d'autres constructions satisfaisant des modèles d'anonymat plus forts ont été proposées. Afin de pouvoir les comparer correctement il est nécessaire de commencer par définir précisément ce qu'est un système de monnaie électronique divisible ainsi que les propriétés de sécurité qu'on attend de lui.

4.2.1 Définition

Un système de monnaie électronique divisible est défini par les algorithmes suivants faisant intervenir trois entités : la banque $\mathcal{B}$, un utilisateur $\mathcal{U}$ et un marchand $\mathcal{M}$.

- $\texttt{Setup}(1^k, V)$: cet algorithme prend en entrée un paramètre de sécurité k et un entier V et retourne les paramètres publics *p.p.* associés à une pièce d'un montant V. On suppose désormais que *p.p.* est pris en entrée par tous les autres algorithmes (y compris par l'adversaire lors des différents jeux de sécurité).
- $\texttt{BKeygen}()$: cet algorithme, qui sera exécuté par la banque $\mathcal{B}$, initialise une liste L et retourne une paire de clés $(\mathsf{bsk}, \mathsf{bpk})$ (on suppose que bsk contient bpk).
- $\texttt{Keygen}()$: cet algorithme, qui sera exécuté par un utilisateur $\mathcal{U}$ (resp. un marchand $\mathcal{M}$), retourne une paire clés $(\mathsf{usk}, \mathsf{upk})$ (resp. $(\mathsf{msk}, \mathsf{mpk})$). On suppose que usk (resp. msk) contient upk (resp. mpk).
- $\texttt{Withdraw}(\mathcal{B}(\mathsf{bsk}, \mathsf{upk}), \mathcal{U}(\mathsf{usk}, \mathsf{bpk}))$: ce protocole interactif s'exécute entre la banque $\mathcal{B}$ et un utilisateur $\mathcal{U}$. À la fin de celui-ci, $\mathcal{U}$ obtient une pièce divisible C de valeur V ou retourne $\perp$ tandis que $\mathcal{B}$ conserve les traces du protocole ou retourne $\perp$.
- $\texttt{Spend}(\mathcal{U}(\mathsf{usk}, C, \mathsf{bpk}, \mathsf{mpk}, v), \mathcal{M}(\mathsf{msk}, \mathsf{bpk}, v))$: ce protocole interactif entre un utilisateur $\mathcal{U}$ et un marchand $\mathcal{M}$ permet au premier de dépenser une valeur v à partir d'une pièce C. Le marchand obtient au final un numéro de série « maître » Z associé à la valeur v ainsi qu'une preuve de validité Π ou retourne $\perp$. L'utilisateur, quant à lui, met à jour sa pièce C ou retourne $\perp$.

– $\texttt{Deposit}(\mathcal{M}(\mathsf{msk}, \mathsf{bpk}, (v, Z, \Pi)), \mathcal{B}(\mathsf{bsk}, L, \mathsf{mpk}))$: ce protocole interactif permet à un marchand de déposer une transaction auprès de la banque. $\mathcal{B}$ vérifie que Π est valide sur v et Z et que (v, z, Π) n'a jamais été déposé. $\mathcal{B}$ retrouve alors m numéros de série $z_1, \ldots, z_m$ correspondant à cette transaction et vérifie si $1 \leq i \leq m$, $z_i \in L$. Si aucun de ces numéros de série ne figure dans L, la banque conserve (v, Z, Π) et rajoute $\{z_1, \ldots, z_m\}$ à sa liste. Autrement, il existe un indice $1 \leq i \leq m$ et un numéro de série $z' \in L$ tel que $z' = z_i$. La banque retrouve alors la transaction (v', Z', Π') correspondante à z' et révèle $[(v, Z, \Pi), (v', Z', \Pi')]$.
– $\texttt{Identify}((v_1, Z_1, \Pi_1), (v_2, Z_2, \Pi_2), \mathsf{bpk})$: Étant données deux traces (v_1, Z_1, Π_1) et (v_2, Z_2, Π_2) de transactions différentes, cet algorithme retourne une clé publique upk ou $\perp$.

Les pièces divisibles C seront le plus souvent associées à des arbres binaires d'une profondeur n définissant $V = 2^n$. Afin d'en faciliter la description, nous introduisons des notations qui seront communes à tous les schémas présentés dans ce chapitre.

Notation. Soit $\mathcal{S}_n$, l'ensemble des chaînes de bits s dont la taille, notée $|s|$, est inférieure ou égale à n (on suppose que la chaîne vide ϵ en fait partie) et $\mathcal{F}_n$ l'ensemble de celles faisant exactement n bits. On définit $\forall s \in \mathcal{S}_n$, l'ensemble $\mathcal{F}_n(s) = \{f \in \mathcal{F}_n : \text{tel que } s \text{ est un préfixe de } f\}$. Pour tout $i \in \{0, \ldots, n\}$, on définit l'ensemble $\mathcal{L}(i) = \{b_{i+1} \ldots b_n : b_j \in \{0, 1\}\}$, c'est-à-dire l'ensemble des chaînes de bits de longueur $n - i$ indexées par $i + 1, \ldots, n$. Ainsi, $\mathcal{L}(n)$ ne contient que la chaîne vide ϵ tandis que $\mathcal{L}(0) = \mathcal{F}_n$.

En pratique, chaque sommet de l'arbre binaire sera désigné par un élément $s \in \mathcal{S}_n$ comme l'illustre la figure 4.1. L'ensemble des feuilles (qui sont donc associées à des éléments de $\mathcal{F}_n$) descendantes d'un sommet s sera donc noté $\mathcal{F}_n(s)$.

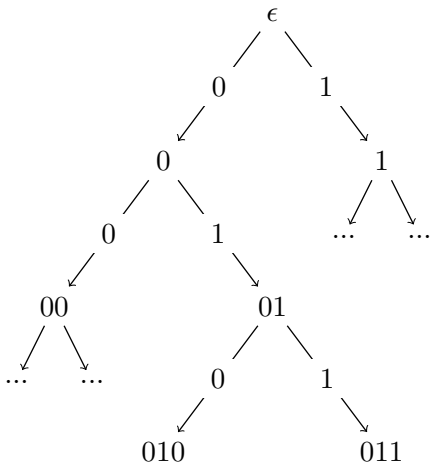


Figure 4.1 – Pièce divisible

4.2.2 Modèle de Sécurité

Outre la propriété classique de complétude, qui signifie informellement qu'un utilisateur honnête sera capable de retirer auprès de la banque une pièce divisible qu'il pourra dépenser chez un marchand, un système de monnaie électronique divisible doit vérifier trois propriétés de sécurité que nous présentons dans cette section. Ces dernières seront modélisées par des jeux de sécurité entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$ ayant accès à certains des oracles suivants.

– $\mathcal{O}\texttt{Add}()$ est un oracle permettant à $\mathcal{A}$ d'enregistrer un nouvel utilisateur (resp. marchand) dans le système. $\mathcal{C}$ exécute alors l'algorithme $\texttt{Keygen}$, conserve usk (resp. msk) dans une liste $\mathcal{HK}$ et renvoie upk (resp. mpk) à $\mathcal{A}$. On dit alors que upk (resp. mpk) est *honnête*.

- $\mathcal{O}\texttt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$ est un oracle utilisé par $\mathcal{A}$ pour corrompre un utilisateur (resp. marchand) honnête dont la clé publique est upk (resp. mpk). $\mathcal{C}$ envoie alors la clé secrète correspondante $\mathsf{usk} \in \mathcal{HK}$ (resp. msk) à $\mathcal{A}$ ainsi que les valeurs secrètes de toutes les pièces retirées par cet utilisateur. On dira alors que upk (resp. mpk) est *corrompu*.
- $\mathcal{O}\texttt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$ est un oracle utilisé par $\mathcal{A}$ pour enregistrer un utilisateur (resp. marchand) corrompu dont la clé publique est upk (resp. mpk). L'adversaire pourrait utiliser cet oracle sur une clé publique déjà enregistrée mais nous préférons interdire ce genre de requêtes pour plus de simplicité. Cela n'entraîne aucune restriction car $\mathcal{A}$ n'y aurait rien gagné de plus qu'en utilisant l'oracle $\mathcal{O}\texttt{Corrupt}$ sur cette même clé publique.
- $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}(\mathsf{upk})$ est un oracle qui exécute la partie utilisateur du protocole de retrait $\texttt{Withdraw}$. Cet oracle sera utilisé par $\mathcal{A}$ jouant le rôle de la banque avec l'utilisateur dont la clé publique est upk.
- $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}(\mathsf{upk})$ est un oracle qui exécute la partie banque du protocole $\texttt{Withdraw}$. Cet oracle sera utilisé par $\mathcal{A}$ jouant le rôle d'un utilisateur dont la clé publique est upk avec la banque.
- $\mathcal{O}\texttt{Spend}(\mathsf{upk}, v)$ est un oracle qui exécute la partie utilisateur du protocole de dépense $\texttt{Spend}$ pour une valeur de v. Cet oracle sera utilisé par $\mathcal{A}$ jouant le rôle d'un marchand $\mathcal{M}$.

Nous désignerons, dans les différents jeux de sécurité, les utilisateurs par leurs clés publiques upk, la valeur dépensée par upk lors des requêtes $\mathcal{O}\texttt{Spend}$ par c_{upk}, et le nombre de pièces divisibles qu'il a retirées par m_{upk} (la valeur totale dont il dispose est donc de $m_{\mathsf{upk}} \cdot V$).

Traçabilité. La traçabilité d'un système de monnaie électronique divisible stipule qu'aucune coalition d'utilisateurs et de marchands ne puisse dépenser plus qu'elle n'a retiré sans que l'un de ses membres ne soit identifié. Elle est formellement définie par le jeu de sécurité $\texttt{Exp}^{tra}_{\mathcal{A}}(1^k, V)$ décrit dans la figure 4.2. L'entier m désigne le nombre total de pièces retirées durant le jeu. On suppose que les traces $(v_1, Z_1, \Pi_1), \ldots, (v_u, Z_u, \Pi_u)$ retournées par $\mathcal{A}$ sont toutes différentes (sinon, il suffit simplement d'éliminer les redondantes). Soit $\texttt{Adv}^{tra}_{\mathcal{A}}(1^k, V) = \texttt{Pr}[\texttt{Exp}^{tra}_{\mathcal{A}}(1^k, V) = 1]$, la probabilité que l'adversaire $\mathcal{A}$ remporte ce jeu. Un système de monnaie électronique divisible sera dit *traçable* si cet avantage est négligeable pour tout adversaire probabiliste polynomial.

$\texttt{Exp}^{tra}_{\mathcal{A}}(1^k, V)$

1. $p.p. \leftarrow \texttt{Setup}(1^k, V)$
2. $(\mathsf{bsk}, \mathsf{bpk}) \leftarrow \texttt{BKeygen}()$
3. $[(v_1, Z_1, \Pi_1), \ldots, (v_u, Z_u, \Pi_u)] \xleftarrow{\$} \mathcal{A}^{\mathcal{O}\texttt{Add}, \mathcal{O}\texttt{Corrupt}, \mathcal{O}\texttt{AddCorrupt}, \mathcal{O}\texttt{Withdraw}_{\mathcal{B}}, \mathcal{O}\texttt{Spend}}(\mathsf{bpk})$
4. Si $\sum_{i=1}^{u} v_i > m \cdot V$ et $\forall i \neq j, \texttt{Identify}((v_i, Z_i, \Pi_i), (v_j, Z_j, \Pi_j)) = \perp$, alors retourner 1
5. Retourner 0

Figure 4.2 – Traçabilité

Remarque 7. Les systèmes de monnaie électronique présentés dans [CG07, ASM08, CG10, IL13] considéraient la notion de *balance*, qui exige qu'il ne soit pas possible de dépenser plus que ce qui a été retiré sans qu'une double-dépense ne soit détectée, et celle d'*identification*, qui empêche toute coalition d'utilisateurs d'effectuer une double-dépense sans qu'une de leurs identités ne soit révélée. La propriété de traçabilité que nous avons introduite recouvre en fait ces deux propriétés. En effet, un adversaire capable de casser la propriété de *balance* pourrait produire $[(v_1, Z_1, \Pi_1), \ldots, (v_u, Z_u, \Pi_u)]$ (avec $\sum_{i=1}^{u} v_i > m \cdot V$) que la banque accepterait comme valide sans détecter de double-dépenses. Il pourrait donc également casser la propriété de traçabilité.

De même, un adversaire contre la propriété d'*identification* doit produire 2 traces (v_1, Z_1, Π_1) et (v_2, Z_2, Π_2) que la banque considèrera comme une double-dépense mais pour lesquelles l'algorithme `Identify` ne pourra retourner d'identité. Il ne reste alors qu'à dépenser (légalement) toutes les pièces restantes pour casser la propriété de traçabilité.

Non-Diffamation. Cette propriété spécifie qu'un utilisateur ne peut être injustement accusé d'avoir effectué une double dépense quand bien même la banque, des utilisateurs et des marchands malhonnêtes se ligueraient contre lui. Elle est formellement définie par le jeu de sécurité $\mathtt{Exp}_{\mathcal{A}}^{ND}(1^k, V)$ décrit dans la figure 4.3.

$\mathtt{Exp}_{\mathcal{A}}^{ND}(1^k, V)$

1. $p.p. \leftarrow \mathtt{Setup}(1^k, V)$
2. $\mathsf{bpk} \leftarrow \mathcal{A}()$
3. $[(v_1, Z_1, \Pi_1), (v_2, Z_2, \Pi_2)] \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{AddCorrupt}, \mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}, \mathcal{O}\mathtt{Spend}}()$
4. Si $\mathtt{Identify}((v_1, Z_1, \Pi_1), (v_2, Z_2, \Pi_2), \mathsf{bpk}) = \mathsf{upk}$ et upk non corrompu, retourner 1
5. Retourner 0

Figure 4.3 – Non-Diffamation

On peut noter que cette propriété implique que les pièces d'un utilisateur honnête ne peuvent être dépensées que par lui. En effet, un adversaire capable de dépenser les pièces d'un utilisateur pourrait, une fois que celui-ci aura tout dépensé, produire une nouvelle dépense qui serait nécessairement détectée comme une fraude. Soit $\mathtt{Adv}_{\mathcal{A}}^{ND}(1^k, V) = \mathtt{Pr}[\mathtt{Exp}_{\mathcal{A}}^{ND}(1^k, V) = 1]$, la probabilité que l'adversaire $\mathcal{A}$ remporte ce jeu. Un système de monnaie électronique divisible vérifiera la propriété de *non-diffamation* si cet avantage est négligeable pour tout adversaire probabiliste polynomial.

Anonymat. Il existe dans la littérature plusieurs variantes de la notion d'anonymat, souvent définies informellement. Nous présentons ici la version la plus forte et précisons ses liens avec certaines autres notions. Idéalement, une transaction ne devrait rien révéler d'autre que les informations nécessaires à son bon déroulement (comme le montant, la date,...). Par ailleurs, un utilisateur ne devrait pas pouvoir être identifié aussi longtemps qu'il est honnête. Tout cela se traduit par le jeu de l'anonymat $\mathtt{Exp}_{\mathcal{A}}^{ano\text{-}b}(1^k, V)$ défini dans la figure 4.4.

L'avantage d'un adversaire $\mathcal{A}$ dans ce jeu est défini par $\mathtt{Adv}_{\mathcal{A}}^{ano}(1^k, V) = \mathtt{Pr}[\mathtt{Exp}_{\mathcal{A}}^{ano\text{-}1}(1^k, V)] - \mathtt{Pr}[\mathtt{Exp}_{\mathcal{A}}^{ano\text{-}0}(1^k, V)]$. Un système de monnaie électronique divisible est *anonyme* si cet avantage est négligeable pour tout adversaire probabiliste polynomial. Les restrictions des étapes 4, 5 et 8 stipulent simplement que les utilisateurs choisis par l'adversaire doivent être honnêtes, enregistrés et surtout qu'ils disposent d'une réserve suffisante pour pouvoir effectuer le paiement demandé. Sans elles, cette distinction serait rendue triviale à cause du mécanisme de détection de double-dépense. Pour les mêmes raisons, l'oracle $\mathcal{O}\mathtt{Spend}$ est remplacé à l'étape 7 par une variante $\mathcal{O}\mathtt{Spend}^*$ qui rejette les requêtes sur (upk_b, v') (pour $b \in \{0, 1\}$) si $c_{\mathsf{upk}_b} + v' > m_{\mathsf{upk}_b} \cdot V - v$.

Remarque 8. Comme nous l'avons expliqué au début de cette section, les premières constructions de monnaie électronique divisible satisfaisaient une propriété de sécurité nettement plus faible où l'utilisateur pouvait être tracé à chaque utilisation d'une même pièce.

Une notion d'anonymat plus forte, appelée *non-corrélation*, fut considérée dans [NS00]. Celle-ci stipule que deux utilisations d'une même pièce ne puissent être reliées. Malheureusement, dans ce modèle, les transactions révèlent malgré tout de l'information sur la pièce, à savoir le sommet dépensé, et nécessitent, en cas de double-dépense, le recours à une autorité de confiance pour

$\mathtt{Exp}_{\mathcal{A}}^{ano\text{-}b}(1^k, V)$

1. $p.p. \leftarrow \mathtt{Setup}(1^k, V)$
2. $\mathsf{bpk} \leftarrow \mathcal{A}()$
3. $(v, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{mpk}) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{Corrupt},\mathcal{O}\mathtt{AddCorrupt},\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}},\mathcal{O}\mathtt{Spend}}()$
4. Si upk_0 ou upk_1 n'est pas enregistré, alors retourner 0
5. Si $c_{\mathsf{upk}_i} > m_{\mathsf{upk}_i} \cdot V - v$ pour $i \in \{0, 1\}$, alors retourner 0
6. $(v, Z, \Pi) \leftarrow \mathtt{Spend}(\mathcal{C}(\mathsf{usk}_b, C, \mathsf{mpk}, v), \mathcal{A}())$
7. $b^* \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{Corrupt},\mathcal{O}\mathtt{AddCorrupt},\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}},\mathcal{O}\mathtt{Spend}^*}()$
8. Si upk_0 ou upk_1 est corrompu, alors retourner 0
9. Retourner $(b = b^*)$

Figure 4.4 – Anonymat

en identifier les auteurs. Bien que cette propriété puisse se révéler suffisante dans certains cas il est important de noter qu'un tel schéma n'est pas anonyme. En effet reprenons l'expérience de sécurité définie dans la figure 4.4. Si l'adversaire connaît le sommet dépensé, il peut très simplement déterminer quel est l'utilisateur impliqué dans la dépense de l'étape 6. Pour cela il lui suffit de faire retirer une seule pièce à upk_0 et upk_1 et d'effectuer $V - 1$ requêtes $\mathcal{O}\mathtt{Spend}(upk_i, 1)$ pour chacun de ces utilisateurs. On peut supposer que le sommet dépensé à chacune de ces requêtes est choisi aléatoirement (si ce n'était pas le cas, il serait encore plus simple d'identifier l'utilisateur) et donc que le dernier sommet restant à upk_0 est différent de celui restant à upk_1. Par conséquent, le sommet utilisé par le challenger dans l'étape 6 révèle immédiatement à l'adversaire quel est l'utilisateur impliqué dans cette dépense.

Nous considérerons donc également dans ce chapitre la notion de *non-corrélation forte* qui spécifie que les transactions ne doivent révéler aucune information sur la pièce dépensée. Un schéma vérifiant cette propriété peut alors être considéré comme anonyme si les auteurs de double-dépense peuvent être identifiés publiquement, c'est-à-dire que le système ne comporte pas d'autorité de confiance capable de lever l'anonymat de n'importe quelle transaction.

4.2.3 État de l'Art

Ainsi définie, la propriété d'anonymat peut sembler très difficile à satisfaire. Pourtant, en 2007, Canard et Gouget [CG07] proposèrent le premier schéma de monnaie électronique divisible la vérifiant. Malheureusement, celui-ci souffre d'une complexité bien trop importante pour pouvoir envisager une utilisation en pratique. Une amélioration de celui-ci, suivant la même idée mais avec plus d'efficacité, fut proposée un peu plus tard par les mêmes auteurs [CG10]. Parallèlement, Au, Susilo et Mu [ASM08] proposèrent une autre construction vérifiant des propriétés de sécurité non conventionnelles. Tous ces schémas sont prouvés sûrs dans le ROM. La seule construction dans le modèle standard était due à Izabachène et Libert [IL13] mais celle-ci ne permet pas, dans la majorité des cas, de retrouver les numéros de série associée à une dépense, ce qui rend la procédure de dépôt à la banque particulièrement compliquée. En effet, chaque fois qu'une transaction est remontée, la banque se retrouve obligée de la comparer (en calculant plusieurs couplages) à chacune des transactions précédemment déposées. Un système de paiement pouvant être amené à supporter des millions de transactions, le coût croissant de l'algorithme `Deposit` va se révéler très rapidement prohibitif. Les auteurs de ce schéma mirent ce défaut sur le compte de l'impossibilité de retrouver les numéros de série qui semblent effectivement indispensables pour détecter simplement les double-dépenses.

En pratique, le cœur d'un système de monnaie électronique divisible se trouve donc être l'arbre binaire modélisant la pièce ou, plus précisément, la manière dont les numéros de série sont construits. Avant de présenter deux nouvelles solutions permettant de construire des systèmes anonymes particulièrement efficaces, même dans le modèle standard, nous commençons par décrire les arbres binaires des schémas existants afin d'en identifier les limitations.

Comme nous l'avons expliqué au début de cette section, une caractéristique de l'arbre binaire associé à une pièce est que les informations relatives à un sommet permettent de retrouver les numéros de série de chacune des feuilles en descendant. Cependant, pour des raisons d'anonymat, il est important que cette procédure fonctionne en sens unique. En effet, s'il était possible, étant donné un numéro de série, de retrouver les informations associées aux sommets dont il descend, on retrouverait les informations de la racine de l'arbre et donc tous les numéros de série. Intuitivement, il semble donc nécessaire que la procédure pour calculer des numéros de série soit basée sur des fonctions à sens unique. Comme nous allons le voir, le choix de cette fonction se révèle fondamental et détermine la plupart des caractéristiques du système de monnaie électronique divisible en résultant.

Le schéma de Canard et Gouget [CG10] (qui est une amélioration de [CG07]) utilise l'exponentiation comme fonction à sens unique. Plus précisément, lorsqu'un utilisateur retire une pièce, il construit récursivement un nouvel arbre en générant un élément $k_\epsilon \in \mathbb{Z}_q^*$ qu'il associe à la racine et en calculant, pour chaque sommet s, l'élément $k_{s||0} \leftarrow g_0^{k_s}$ associé au fils gauche et l'élément $k_{s||1} \leftarrow g_1^{k_s}$ associé au fils droite (g_0 et g_1 étant des éléments d'ordre p de $\mathbb{Z}_q^*$ où $p|(q-1)$). A priori, cette construction semble poser problème car k_s est vu à la fois comme un élément de $\mathbb{Z}_q^*$ et comme un élément de $\mathbb{Z}_p$. Cependant, cela n'est dû qu'à un abus de notation, les éléments $k_{s||0}$ et $k_{s||1}$ étant en fait définis comme $g_0^{[k_s]}$ et $g_1^{[k_s]}$ où $[k_s]$ est l'un des entiers (dont le choix devra être spécifié dans les paramètres du système) tel que $[k_s] = k_s \bmod q$. Ainsi, étant donné k_s associé à un sommet s, il est possible, par exponentiations successives, de retrouver chacun des numéros de série dérivés (qui sont les éléments k_f pour $f \in \mathcal{F}_n(s)$). Malheureusement, ce choix de construction pose deux problèmes. Tout d'abord, cela impose de travailler avec des sous-groupes de corps finis premiers car eux seuls bénéficient d'un lien aussi explicite entre leurs éléments et les entiers. En effet, si k_s était un point d'une courbe elliptique, l'élévation à la puissance k_s n'aurait alors plus aucun sens. Cela interdit donc l'utilisation de courbes elliptiques qui offrent, comme nous l'avons vu dans le chapitre 2, une efficacité nettement supérieure à celle des corps finis. L'autre problème survient lorsque l'utilisateur souhaite prouver la validité de son arbre. Cette étape, qui est indispensable pour éviter que l'utilisateur ne triche lors de la construction de son arbre (ce qui lui permettrait de dépenser beaucoup plus qu'il n'a retiré), consiste, lors d'une dépense, à prouver que $k_s = g_i^{k_{s'}}$ pour un certain $i \in \{0,1\}$ et s' parent de s. L'élément $k_{s'} \in \mathbb{Z}_q$ ne pouvant être révélé, l'utilisateur va devoir en prouver la connaissance et montrer qu'il est certifié. Si cette preuve ne pose pas de problème pour des groupes de même ordre, il n'en va pas de même ici où $k_{s'}$ est vu successivement comme un élément de $\mathbb{Z}_q^*$ et comme un élément de $\mathbb{Z}_p$. En effet, le protocole de Schnorr étant incompatible avec cette situation, il est nécessaire d'avoir recours à d'autres protocoles bien plus coûteux (par exemple [BT99]) dont la sécurité repose sur le ROM.

Ainsi, l'utilisation de l'exponentiation comme fonction à sens unique impose le recours à des groupes d'ordre différent, ce qui augmente la complexité générale du protocole et entraîne une preuve de sécurité dans le ROM.

Le schéma [ASM08] proposé par Au, Susilo et Mu combine pour sa part l'exponentiation et une fonction de hachage cryptographique afin de construire les numéros de série. Plus précisément, après avoir sélectionné un élément $k_\epsilon \in \mathbb{Z}_p$ associé à la racine, l'utilisateur va générer le reste de l'arbre en calculant, pour chaque $s \in \mathcal{S}_n$, les éléments $k_{s||0} \leftarrow \mathcal{H}_0(g^{k_s})$ et $k_{s||1} \leftarrow \mathcal{H}_1(g^{k_s})$ (où g est un générateur d'un groupe cyclique $\mathbb{G}$ d'ordre p et $\mathcal{H}_0$, $\mathcal{H}_1 : \mathbb{G} \to \mathbb{Z}_p$ sont deux fonctions de hachage). Cette construction évite donc le problème rencontré par les schémas de Canard et Gouget car la fonction de hachage permet de définir k_s comme un élément de $\mathbb{Z}_p$, quelle

que soit la nature du groupe $\mathbb{G}$. En pratique, il est donc possible d'utiliser le groupe de points d'une courbe elliptique et donc de bénéficier de meilleures performances. Malheureusement, le recours à des fonctions de hachages cryptographiques n'est pas sans conséquence. En effet, il devient alors impossible de prouver que k_s est bien formé à l'aide de preuves de connaissance, ces dernières n'étant pas compatibles avec ces fonctions. Pour pallier ce défaut, les auteurs proposèrent un système de type *cut-and-choose* où la banque peut demander, lors d'un retrait, à obtenir tous les informations de l'arbre afin d'en vérifier la validité. Si l'arbre est valide, l'utilisateur devra à nouveau générer un arbre. Sinon, la banque lui inflige une amende pour avoir tenté de tricher. L'argument des auteurs est que, si l'amende est suffisamment élevée, alors des résultats classiques de probabilités garantissent que la banque ne perdra pas d'argent « en moyenne ». Malheureusement, ce genre de solution n'est pas réaliste en pratique, d'une part parce qu'il est peu probable qu'une banque accepte ce modèle économique correspondant davantage à un casino et d'autre part parce qu'il semble peu évident, d'un point de vue légal, que la banque puisse imposer des amendes à ses clients. Par exemple, un utilisateur malhonnête pourrait les contester en soutenant que la malformation de son arbre est simplement dû à un « bug » de son périphérique.

Là encore, les problèmes dont souffrent ce schéma semblent inhérents aux choix de la fonction à sens unique permettant de calculer les numéros de série. Nous proposons dans les prochaines sections deux nouvelles constructions d'arbre qui permettent des constructions nettement plus efficaces, y compris dans le modèle standard.

4.3 Une Nouvelle Construction

Nous présentons dans cette section un nouveau schéma de monnaie électronique divisible issu de l'article *Divisible E-Cash Made Practical* [CPST15a] publié à la conférence PKC 2015.

4.3.1 Idée Générale

Comme nous l'avons expliqué dans la section précédente, un schéma de monnaie électronique divisible dépend essentiellement de la manière dont les numéros de série sont construits. Notre nouvelle construction se distingue des précédentes sur deux points.

Tout d'abord, la fonction à sens unique utilisée pour calculer les numéros de série est ici le couplage bilinéaire. Chaque fois qu'un utilisateur dépensera un sommet s il va en effet révéler un élément $t_s \in \mathbb{G}_1$ qui permettra à la banque de retrouver tous les numéros de série dérivés en calculant $e(t_s, \widetilde{g}_{s \mapsto f})$ pour chaque $f \in \mathcal{F}_n(s)$ (les éléments $\widetilde{g}_{s \mapsto f}$ figurant dans les paramètres publics du système). Une des conséquences de l'utilisation des couplages est que notre système est maintenant défini dans un environnement bilinéaire qui est compatible avec les preuves Groth-Sahai (présentées au chapitre 3), ce qui va permettre de l'instancier dans le modèle standard.

La deuxième nouveauté, rendue possible par l'utilisation du couplages, est qu'il va désormais exister une unique structure d'arbre, commune à tous les utilisateurs. Cela va conduire à une augmentation significative de l'efficacité des protocoles de dépense (`Spend`) et de retrait (`Withdraw`), les rendant exécutables en temps constant. En effet, dans les constructions précédentes, chaque utilisateur doit, lors d'un retrait, générer un nouvel arbre en calculant des éléments k_s pour chaque sommet, puis les faire certifier. En pratique, cela se fait par le biais d'accumulateurs (un par niveau) qui permettent à l'utilisateur de limiter le nombre de certificats à conserver, mais dont la complexité du calcul reste malgré tout linéaire en le nombre de sommets. Par ailleurs, l'utilisateur doit également (lors du retrait [ASM08] ou lors de la dépense [CG10]) prouver que son arbre est bien formé ce qui, là encore, augmente sensiblement la complexité du schéma.

Pour éviter cela, notre système va définir, dans les paramètres publics, une unique structure d'arbre T que les utilisateurs n'auront donc pas besoin de faire certifier et dont la validité pourra être vérifiée une fois pour toutes lors du `Setup`. Plus précisément, chaque feuille de T sera associée

à un élément $\chi_f \in \mathbb{G}_T$ et chaque sommet s à un élément $g_s \in \mathbb{G}_1$. Par ailleurs, les paramètres publics contiendront également, pour chaque $s \in \mathcal{S}_n$ et chaque $f \in \mathcal{F}_n(s)$, un élément $\widetilde{g}_{s \mapsto f} \in \mathbb{G}_2$ (dont la construction sera explicitée dans la prochaine section) tel que $e(g_s, \widetilde{g}_{s \mapsto f}) = \chi_f$.

Bien évidemment, le fait que la structure d'arbre soit commune ne signifie pas que tous les utilisateurs vont disposer du même arbre. Au contraire, il est important que chaque nouvelle pièce définisse un nouvel arbre. Pour cela, l'utilisateur va, lors d'un retrait, obtenir un certificat sur un scalaire $x \in \mathbb{Z}_p$ de son choix qui va définir implicitement les numéros de série comme étant χ_f^x pour chaque $f \in \mathcal{F}_n$. Cette stratégie conduit à un protocole de retrait très efficace, notamment parce que l'utilisateur n'a pas besoin de générer l'arbre entier, mais permet également une très grande modularité dans le protocole de dépense qui peut être ajusté en fonction du niveau d'anonymat souhaité. En effet, notre construction, qui est initialement prévue pour être non-corrélable, peut être modifiée pour atteindre des propriétés de sécurité plus fortes, telles que la non-corrélation forte et l'anonymat.

Construction d'un schéma non-corrélable. À la fin du protocole de retrait, l'utilisateur détient donc un scalaire x ainsi qu'un certificat dessus. Pour dépenser une valeur 2^l, il va sélectionner un sommet s de niveau $n - l$ non dépensé (c'est-à-dire dont aucun des numéros de série dérivés n'a été révélé) et calculer $t_s \leftarrow g_s^x$. Il doit ensuite prouver que cet élément est bien formé et que x est certifié, ce qui peut se faire très simplement, tant dans le ROM que dans le modèle standard.

Le schéma étant simplement non-corrélable, l'utilisateur peut révéler le sommet s qu'il a utilisé, ce qui permet à la banque de retrouver les éléments $S_f(x) \leftarrow e(t_s, \widetilde{g}_{s \mapsto f}) = e(g_s, \widetilde{g}_{s \mapsto f})^x = \chi_f^x$ qui sont en fait les 2^l numéros de série. Le point important est que ces éléments ne dépendent que du secret x de la pièce et de la feuille f, ce qui rend possible la détection de double-dépenses. Celles-ci consistent en effet en (au moins) deux dépenses de sommets s et s' disposant d'une feuille f commune. L'égalité $e(t_s, \widetilde{g}_{s \mapsto f}) = \chi_f^x = e(t_{s'}, \widetilde{g}_{s' \mapsto f})$ implique alors une collision entre les numéros de série des deux dépenses et donc une détection de la fraude.

Pour les utilisateurs honnêtes, la propriété de non-corrélation provient simplement de la difficulté de décider, étant donnés les paramètres publics (dont g_s et $g_{s'}$) et g_s^x, si un élément de $\mathbb{G}_1$ est aléatoire ou égal à $g_{s'}^x$, ce qui correspond à l'hypothèse $\mathsf{EXDH} - 1$.

Construction d'un schéma fortement non-corrélable. Supposons maintenant que l'on souhaite renforcer le niveau d'anonymat du système pour atteindre la propriété de non-corrélation forte. Il devient alors nécessaire de masquer toutes les informations concernant le sommet dépensé. Si révéler t_s reste possible (car identifier l'élément g_s utilisé revient également à résoudre le problème $\mathsf{EXDH} - 1$), en prouver la validité devient plus compliqué. En effet l'utilisateur doit désormais prouver qu'il a utilisé un élément g_s du niveau correspondant au montant sans le révéler.

Une solution classique consiste à rajouter dans la clé publique de la banque un certificat (délivré par la banque) sur chacun des g_s sous une clé publique dépendant du niveau $|s|$. Lors d'une dépense, l'utilisateur pourra prouver connaissance d'un tel certificat ce qui assurera que l'élément t_s a été bien formé.

Cependant, un autre problème se pose lors du protocole de dépôt. En effet, la banque ne connait plus le sommet s utilisé et ne sait donc pas quels éléments $\widetilde{g}_{s \mapsto f}$ choisir pour retrouver les numéros de série. Comme $|s|$ est connu (car il correspond au montant de la dépense), elle va calculer, pour chaque sommet s' de ce niveau et chaque $f \in \mathcal{F}_n(s)$, les éléments $e(t_s, \widetilde{g}_{s' \mapsto f})$. En procédant de la sorte, la banque est assurée de retrouver tous les numéros de série (et donc de pouvoir continuer à détecter les double-dépenses) mais va également se retrouver avec de nombreux éléments (à savoir les $e(t_s, \widetilde{g}_{s' \mapsto f})$ pour $s' \neq s$) qui ne seront pas valides.

Cette solution, qui permet de garantir toutes les propriétés de sécurité du schéma, entraîne donc un surcout significatif lors du protocole de dépôt. Nous verrons dans la section suivante une autre façon de construire l'arbre qui permet d'éviter ce problème.

Construction d'un schéma anonyme. Si la construction présentée jusque-là (que sa non-corrélation soit forte ou non) permet de détecter une double-dépense, nous n'avons pas encore décrit comment en identifier les auteurs. Il existe en fait deux approches très différentes.

La première consiste à confier à une certaine autorité, comme pour les signatures de groupe, le pouvoir d'identifier l'auteur de n'importe quelle transaction. Si cette solution ne soulève en général aucun problème technique, il n'en reste pas moins qu'elle rentre un peu en contradiction avec l'objectif de départ de la monnaie électronique, à savoir protéger l'anonymat des utilisateurs honnêtes. Ce type de système de monnaie électronique divisible, dit *équitable*, peut donc être critiqué car il ne fait que transférer la capacité de tracer les utilisateurs de la banque à une certaine entité en laquelle les utilisateurs auront plus confiance. Cependant, en pratique, cela peut se révéler nécessaire pour des raisons légales, l'entité en question pouvant par exemple être une autorité judiciaire.

La deuxième approche, correspondant à la propriété d'anonymat définie dans la section 4.2.2, permet une levée d'anonymat publique mais seulement dans le cas d'une double-dépense. Pour proposer cette fonctionnalité, le schéma fortement non-corrélable que nous venons de présenter doit être modifié comme suit.

Les paramètres publics sont augmentés d'un élément $h \leftarrow g^c$ pour un certain $c \xleftarrow{\$} \mathbb{Z}_p$ ainsi que des éléments $h_s \leftarrow g_s^c$ pour tout $s \in \mathcal{S}_n$. Soit $\mathsf{usk} \in \mathbb{Z}_p$ la clé secrète d'un utilisateur et $\mathsf{upk} \leftarrow g^{\mathsf{usk}}$ la clé publique correspondante. Lors d'une dépense, l'utilisateur va, en plus de t_s, calculer $v_s \leftarrow \mathsf{upk}^R \cdot h_s^x$, où R est obtenu en hachant certaines données relatives à la transaction (telles que le montant, la date, etc...), et en prouver la validité. La paire (t_s, v_s) qu'il va envoyer peut donc être vue comme un chiffré El Gamal [ElG84] de la clé publique upk de l'utilisateur élevée à une puissance R connue. Le point important est que ce chiffré est généré de manière déterministe en fonction du sommet s. Tant que l'utilisateur sera honnête, le « masque » h_s^x ne sera utilisé qu'une seule fois et assurera sa confidentialité. A l'inverse, un fraudeur n'aura pas d'autre choix que de réutiliser un masque, ce qui va permettre de l'identifier.

En effet, lorsqu'une double dépense sera détectée, la banque disposera de deux traces de transactions contenant (t_s, v_s) et $(t_{s'}, v_{s'})$ où s et s' sont deux sommets partageant au moins une feuille commune $f \in \mathcal{F}_n(s) \cap \mathcal{F}_n(s')$. Par conséquent, nous avons $e(h_s, \widetilde{g}_{s\mapsto f}) = e(h_{s'}, \widetilde{g}_{s'\mapsto f})$ ainsi que l'égalité suivante (où $T = e(v_s, \widetilde{g}_{s\mapsto f})$ et $T' = e(v_{s'}, \widetilde{g}_{s'\mapsto f})$) :

$$\begin{aligned} T \cdot T'^{-1} &= e(\mathsf{upk}^R, \widetilde{g}_{s\mapsto f}) \cdot e(h_s^x, \widetilde{g}_{s\mapsto f}) \cdot e(\mathsf{upk}^{-R'}, \widetilde{g}_{s'\mapsto f}) \cdot e(h_{s'}^x, \widetilde{g}_{s'\mapsto f})^{-1} \\ &= e(\mathsf{upk}, \widetilde{g}_{s\mapsto f}^R \cdot \widetilde{g}_{s'\mapsto f}^{-R'}). \end{aligned}$$

L'élément $T \cdot T'^{-1}$ ne dépend donc que de upk et de paramètres connus des transactions. Il est alors possible de tester, pour toute clé publique upk_i du système, si $T \cdot T'^{-1} = e(\mathsf{upk}_i, \widetilde{g}_{s\mapsto f}^R \cdot \widetilde{g}_{s'\mapsto f}^{-R'})$ jusqu'à tomber sur une égalité.

L'algorithme d'identification (`Identify`) a donc un coût linéaire en le nombre d'utilisateurs, ce qui n'est pas vraiment problématique car les cas de fraudes restent rares notamment en raison de l'utilisation de périphériques sécurisés (tels que des cartes à puces). Par ailleurs, cette procédure peut être efficacement parallélisée et déléguée comme c'est le cas pour le schéma de traçage de traîtres décrit dans [CPP05].

4.3.2 Construction

Nous décrivons à présent, de manière plus formelle, les algorithmes constituant notre système de monnaie électronique divisible anonyme en distinguant le cas d'une construction dans le ROM de celui d'une construction dans le modèle standard. Cette distinction est rendue nécessaire par le fait que les preuves Groth-Sahai ne permettent pas de prouver efficacement la connaissance de scalaires, contrairement aux preuves de connaissance de type Schnorr. Dans le premier cas

l'utilisateur devra donc prouver la connaissance d'un élément $u_2^x \in \mathbb{G}_1$ (pour un certain paramètre public u_2), tandis que dans le deuxième cas, prouver la connaissance de x suffira.

- $\texttt{Setup}(1^k, V)$: soient $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires d'ordre p et g, h, u_1, u_2, w (resp. $\widetilde{g}$) des générateurs de $\mathbb{G}_1$ (resp. $\mathbb{G}_2$). L'entité générant les paramètres du système (voir remarque 9) sélectionne une fonction de hachage $H : \{0,1\}^* \to \mathbb{Z}_p$ résistante aux collisions et construit les éléments suivants :
  - pour tout $s \in \mathcal{S}_n$, $r_s \xleftarrow{\$} \mathbb{Z}_p$ et $(g_s, h_s) \leftarrow (g^{r_s}, h^{r_s})$;
  - pour tout $f \in \mathcal{F}_n$, $l_f \xleftarrow{\$} \mathbb{Z}_p$;
  - pour tout $s \in \mathcal{S}_n$, pour tout $f \in \mathcal{F}_n(s)$, $\widetilde{g}_{s \mapsto f} \leftarrow \widetilde{g}^{l_f / r_s}$.

  Les paramètres publics du système sont alors définis comme étant $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, ainsi que g, h, u_1, u_2, w, $\widetilde{g}$, et $G = e(g, \widetilde{g})$, la fonction H, et les ensembles $\{(g_s, h_s), s \in \mathcal{S}_n\}$ et $\{\widetilde{g}_{s \mapsto f}, s \in \mathcal{S}_n, f \in \mathcal{F}_n(s)\}$. Par ailleurs, en fonction du modèle de sécurité, on rajoute :
  - une autre fonction de hachage $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$, qui sera modélisée par un oracle aléatoire ;
  - ou une chaîne de référence commune (CRS) pour le système de preuves Groth-Sahai ainsi que la description d'un schéma de signature à usage unique Σ_{ots} tel que celui proposé dans [BB08].

  Il est important de noter que les utilisateurs et les marchands ont seulement besoin de connaître les différents générateurs et $\{(g_s, h_s), s \in \mathcal{S}_n\}$ (ainsi que $\mathcal{H}$, ou CRS et Σ_{ots}). En pratique, l'ensemble $\{\widetilde{g}_{s \mapsto f}, (s, f) \in \mathcal{S}_n \times \mathcal{F}_n\}$ sera seulement utilisé par la banque, tandis que $\{l_f, f \in \mathcal{F}_n\}$ ne sera plus d'aucune utilité dans le système.
- $\texttt{BKeygen}()$: après avoir reçu les paramètres du système, la banque va sélectionner deux schémas de signature.
  - $\Sigma_0 = (\texttt{Keygen}, \texttt{Sign}, \texttt{Verify})$, dont l'espace des message est $\mathbb{G}_1^2$, pour signer certains éléments des paramètres publics. Un choix possible est le schéma de signature proposé dans [AGHO11] bien qu'il vérifie une propriété de sécurité (EUF-CMA) nettement plus forte que ce dont nous avons besoin. En pratique, un schéma sûr face à des attaques sélectives à messages choisis serait suffisant.
  - $\Sigma_1 = (\texttt{Keygen}, \texttt{Sign}, \texttt{Verify})$, dont l'espace des messages dépend du modèle de sécurité.
    - **ROM :** l'espace des messages est ici $\mathbb{Z}_p^2$. Il est cependant nécessaire que le schéma choisi soit compatible avec un protocole $\Sigma_1.\texttt{SignCommit}$ qui, étant donné (u_1^x, u_2^y) pour une certaine paire $(x, y) \in \mathbb{Z}_p^2$ (donc une mise en gage de (x, y)), retourne une signature valide σ sur (x, y). C'est par exemple le cas des schémas dûs à Camenisch et Lysyanskaya et à Boneh et Boyen que nous avons présentés dans la section 3.2.2 ou celui que nous décrivons dans le chapitre 8.
    - **Modèle Standard :** l'espace des messages est ici $\mathbb{G}_1^2$, ce qui rend de nouveau possible l'utilisation du schéma de [AGHO11].

  Dans tous les cas, la banque obtient $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \Sigma_1.\texttt{Keygen}(p.p.)$ ainsi que $(\mathsf{sk}_0^{(i)}, \mathsf{pk}_0^{(i)}) \leftarrow \Sigma_0.\texttt{Keygen}(p.p.)$ pour chaque niveau i de l'arbre $(0 \leq i \leq n)$ et calcule, pour tout $s \in \mathcal{S}_n$, la signature $\tau_s \leftarrow \Sigma_0.\texttt{Sign}(\mathsf{sk}_0^{(|s|)}, (g_s, h_s))$. Finalement, il définit bsk comme étant sk_1 et bpk comme étant $(\{\mathsf{pk}_0^{(i)}\}_i, \mathsf{pk}_1, \{\tau_s\}_{s \in \mathcal{S}_n})$.
- $\texttt{Keygen}()$: chaque utilisateur (resp. marchand) choisit une clé secrète aléatoire $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p$ (resp. msk) et calcule $\mathsf{upk} \leftarrow g^{\mathsf{usk}}$ (resp. $\mathsf{mpk} \leftarrow g^{\mathsf{msk}}$). Dans ce qui suit, nous supposerons que upk (resp. mpk) est publiquement accessible, c'est-à-dire que tout le monde peut en obtenir une copie valide.
- $\texttt{Withdraw}(\mathcal{B}(\mathsf{bsk}, \mathsf{upk}), \mathcal{U}(\mathsf{usk}, \mathsf{bpk}))$: pour retirer une pièce divisible, l'utilisateur commence par générer un scalaire $x \in \mathbb{Z}_p$ aléatoire, puis calcule u_1^{usk} et u_2^x. Il envoie alors $\mathsf{upk}, u_1^{\mathsf{usk}}$ et u_2^x à la banque et lui prouve, en utilisant par exemple une extension du protocole interactif de Schnorr, sa connaissance de x et usk. Si cette preuve est valide et que u_2^x n'a pas déjà

été utilisé, la banque
- **ROM :** exécute le protocole $\Sigma_1.\texttt{SignCommit}$ sur $(u_1^{\mathsf{usk}}, u_2^x)$ et envoie la signature résultante σ à l'utilisateur qui définit $C \leftarrow (x, \sigma)$.
- **Modèle Standard :** exécute l'algorithme $\sigma \leftarrow \Sigma_1.\texttt{Sign}(\mathsf{sk}_1, (u_1^{\mathsf{usk}}, u_2^x))$ et retourne σ à l'utilisateur qui définit $C \leftarrow (x, \sigma)$.

– $\texttt{Spend}(\mathcal{U}(\mathsf{usk}, C, \mathsf{bpk}, \mathsf{mpk}, 2^\ell), \mathcal{M}(\mathsf{msk}, \mathsf{bpk}, 2^\ell))$: pour dépenser une valeur 2^ℓ, l'utilisateur sélectionne un sommet non dépensé s de niveau $n - \ell$ et calcule $R \leftarrow H(info)$ (où $info$ est un ensemble de données publiques associées au paiement, telles que le montant, la date, etc...) ainsi que $(t_s, v_s) \leftarrow (g_s^x, \mathsf{upk}^R \cdot h_s^x)$. Il doit alors prouver que t_s et v_s sont bien formés, c'est-à-dire qu'il a utilisé des valeurs préalablement certifiées lors d'un retrait, donc une preuve de connaissance de σ, et qu'il a utilisé une paire valide (g_s, h_s), donc une preuve de l'existence de τ_s. Là encore, le protocole dans le ROM diffère de celui dans le modèle standard.
  - **ROM :** l'utilisateur fournit une preuve de connaissance de usk, x, g_s, h_s, τ_s, et σ vérifiant :

$$t_s = g_s^x \wedge v_s = (g^R)^{\mathsf{usk}} \cdot h_s^x \quad \wedge \quad \Sigma_1.\texttt{Verify}(\mathsf{pk}_1, (\mathsf{usk}, x), \sigma) = 1$$
$$\wedge \quad \Sigma_0.\texttt{Verify}(\mathsf{pk}_0^{(n-\ell)}, (g_s, h_s), \tau_s) = 1.$$

    Cette preuve est alors convertie en une signature de connaissance Π sur le message R, en utilisant la méthode Fiat-Shamir présentée dans la section 3.3.3.
  - **Modèle Standard :** l'utilisateur doit fournir une preuve Groth-Sahai de connaissance de σ et τ_s. Cependant, l'utilisation de ce type de preuves pose un problème car elles peuvent être régénérées, c'est-à-dire qu'étant donnée une preuve valide π, il est possible, sans la connaissance des secrets, d'en calculer une nouvelle version $\tilde{\pi} \neq \pi$ prouvant les mêmes affirmations. Dans notre cas, un marchand malhonnête pourrait régénérer la preuve d'une trace valide et la déposer à nouveau. Un utilisateur serait alors injustement accusé de fraude, ce qui contredirait la propriété de non-diffamation attendue d'un système de monnaie électronique divisible.
    Pour éviter cela, l'utilisateur va générer une paire de clés pour un schéma de signature à usage unique $(\mathsf{sk}_{ots}, \mathsf{pk}_{ots}) \leftarrow \Sigma_{ots}.\texttt{Keygen}(1^k)$ et certifier la clé publique en calculant $\mu \leftarrow w^{\frac{1}{\mathsf{usk}+H(\mathsf{pk}_{ots})}}$. La clé secrète sera ensuite utilisée pour signer l'intégralité de la trace (et donc les preuves). Une régénération des preuves donne lieu à une nouvelle trace qui nécessite donc une nouvelle signature. Par conséquent, l'attaque précédente ne s'applique plus car aucune entité, à part l'utilisateur, n'est capable de la produire.
    Le calcul de la preuve Groth-Sahai commence par les mises en gages de usk, x, g_s, h_s, τ_s, σ, μ, $U_1 = u_1^{\mathsf{usk}}$ et $U_2 = u_2^x$. L'utilisateur fournit ensuite une preuve non-interactive à divulgation nulle π que celles-ci vérifient :

$$t_s = g_s^x \wedge v_s = (g^R)^{\mathsf{usk}} \cdot h_s^x \wedge U_2 = u_2^x \wedge U_1 = u_1^{\mathsf{usk}} \wedge \mu^{(\mathsf{usk}+H(\mathsf{pk}_{ots}))} = w$$

    ainsi qu'une preuve non-interactive π' satisfaisant l'indistinguabilité des témoins que les valeurs mises en gage vérifient :

$$1 = \Sigma_0.\texttt{Verify}(\mathsf{pk}_0^{(n-\ell)}, (g_s, h_s), \tau_s) \wedge 1 = \Sigma_1.\texttt{Verify}(\mathsf{pk}_1, (U_1, U_2), \sigma)$$

    Finalement, l'utilisateur calcule $\eta \leftarrow \Sigma_{ots}.\texttt{Sign}(\mathsf{sk}_{ots}, H(t_s||v_s||\pi||\pi'||R))$ et l'envoie à $\mathcal{M}$ ainsi que $t_s, v_s, \mathsf{pk}_{ots}, \pi, \pi'$.

Dans tous les cas, le marchand vérifie que les preuves et les signatures sont valides auquel cas il accepte la transaction. Il conserve alors $Z \leftarrow (t_s, v_s)$ et soit la signature de connaissance Π (dans le cas du ROM) soit l'ensemble $\Pi \leftarrow (\pi, \pi', \mathsf{pk}_{ots}, \eta)$ (dans le cas du modèle standard).

– $\texttt{Deposit}(\mathcal{M}(\mathsf{msk}, \mathsf{bpk}, (2^\ell, Z, \Pi)), \mathcal{B}(\mathsf{bsk}, L, \mathsf{mpk}))$: lorsqu'un marchand fait remonter la trace d'un paiement à la banque, celle-ci va commencer par vérifier qu'elle n'a pas déjà été remontée et qu'elle est valide. Si c'est le cas, elle va, pour chaque s' de niveau $n - \ell$ et $f \in \mathcal{F}_n(s')$, calculer $z_i \leftarrow e(t_s, \widetilde{g}_{s' \mapsto f})$. Elle vérifie ensuite que $\forall i, z_i \notin L$ auquel cas elle ajoute ces éléments à cette liste (voir la remarque ci-dessous) et conserve la trace $(2^\ell, Z, \Pi)$. Sinon, il existe un élément $z' \in L$ tel que $z_i = z'$. La banque retrouve alors la trace $(2^{\ell'}, Z', \Pi')$ correspondante et retourne $[(2^\ell, Z, \Pi), (2^{\ell'}, Z', \Pi')]$.

– $\texttt{Identify}((2^{\ell_1}, Z_1, \Pi_1), (2^{\ell_2}, Z_2, \Pi_2), \mathsf{bpk})$: pour identifier l'auteur d'une double dépense, l'entité exécutant cet algorithme va commencer par vérifier la validité des deux traces et retourne $\perp$ si l'une d'entre elles n'est pas correcte. Il calcule alors, pour $i \in \{1, 2\}$ et pour toute feuille f, les listes $S_{i,f} \leftarrow \{e(t_{s_i}, \widetilde{g}_{s \mapsto f}), \forall s \in \mathcal{S}_n$ tel que $|s| = |s_i|$ et $f \in \mathcal{F}_n(s)\}$, et retourne $\perp$ s'il n'y a aucune collision entre $S_{1,f}$ et $S_{2,f}$ pour toute feuille f. Sinon, on peut supposer (voir la remarque ci-dessous) que nous avons $e(t_{s_1}, \widetilde{g}_{s_1 \mapsto f}) = e(t_{s_2}, \widetilde{g}_{s_2 \mapsto f})$ avec $t_{s_1} = g_{s_1}^x$ et $t_{s_2} = g_{s_2}^x$ pour des sommets $s_1, s_2 \in \mathcal{S}_n$. Comme expliqué dans la section précédente, on a alors la relation $e(v_{s_1}, \widetilde{g}_{s_1 \mapsto f}) \cdot e(v_{s_2}, \widetilde{g}_{s_2 \mapsto f})^{-1} = e(\mathsf{upk}, \widetilde{g}_{s \mapsto f}^R \cdot \widetilde{g}_{s' \mapsto f}^{-R'})$. Il ne reste donc plus qu'à calculer $e(\mathsf{upk}_i, \widetilde{g}_{s \mapsto f}^R \cdot \widetilde{g}_{s' \mapsto f}^{-R'})$ pour toutes les clés publiques jusqu'à avoir une correspondance, auquel cas on retourne upk_i. On peut remarquer que la propriété de résistance aux collisions attendue de H est nécessaire. Sans celle-ci, on pourrait avoir $R = R'$, et donc $\widetilde{g}_{s \mapsto f}^R \cdot \widetilde{g}_{s' \mapsto f}^{-R'} = 1_{\mathbb{G}_2}$ dans le cas où $s = s'$, rendant impossible toute identification.

Remarque 9.

1. Nous avons considéré, pour plus de simplicité, que l'algorithme `Setup` était exécuté par une unique entité. Cependant, la connaissance des scalaires (r_s, l_f) utilisés dans cette phase permet de casser l'anonymat du schéma. La génération de la CRS des preuves Groth-Sahai pose le même problème. En pratique, il est donc indispensable que les paramètres du système soient générés par une autorité de confiance ou alors coopérativement par différentes entités aux intérêts divergents (telles que la banque et un ensemble d'utilisateurs). Par exemple, la banque pourrait générer $a_s, c_f \xleftarrow{\$} \mathbb{Z}_p$ pour tout $s \in \mathcal{S}_n$ et $f \in \mathcal{F}_n$, calculer $A_s \leftarrow g^{a_s}$ et $\widetilde{A}_{s \mapsto f} \leftarrow \widetilde{g}^{c_f / a_s}$, et prouver sa connaissance de a_s et c_f. Les utilisateurs pourraient alors choisir des scalaires aléatoires b_s et d_f, calculer $B_s \leftarrow A_s^{b_s}$ ainsi que $\widetilde{B}_{s \mapsto f} \leftarrow \widetilde{A}_s^{d_f / b_s}$ et prouver connaissance de b_s et d_f.

   Si toutes les preuves sont valides alors les paramètres publics sont définis par $g_s \leftarrow B_s$ et $\widetilde{g}_{s \mapsto f} \leftarrow \widetilde{B}_{s \mapsto f}$ dont les logarithmes discrets ne sont connus par aucune entité.

2. Le sommet impliqué dans une dépense n'étant pas connu, la banque doit calculer et stocker 2^n éléments z_i à chaque fois qu'une transaction lui est remontée, peu importe son montant. Dans le pire des cas (si l'utilisateur dépense sa pièce centime par centime) la banque se retrouvera obligée de conserver 2^{2n} éléments pour chaque pièce d'une valeur 2^n. Cependant, en pratique, la banque n'a pas besoin de stocker les éléments $z_i \in \mathbb{G}_T$ mais seulement leur empreinte (nettement plus petite) $H'(z_i)$ pour une certaine fonction de hachage H'. En cas de collision, la banque commencera par recalculer les z_i pour être sûre que celle-ci n'est pas due à la fonction H' avant de lancer la procédure d'identification.

3. Le risque de cette construction, où la liste L contient un grand nombre de numéros de série invalides, serait la présence de faux positifs, à savoir deux pièces de secrets respectifs x_1 et x_2 pour lesquelles il existerait des sommets s_1, s_1', s_2 et s_2' tels que $e(g^{r_{s_1} x_1}, \widetilde{g}^{y_f / r_{s_1'}}) = e(g^{r_{s_2} x_2}, \widetilde{g}^{y_f / r_{s_2'}})$ pour un certain $f \in \mathbb{F}_n$. Cependant, cela impliquerait que $x_1 r_{s_1} r_{s_2'} = x_2 r_{s_2} r_{s_1'}$ ce qui n'arrive qu'avec une probabilité négligeable lorsque x_1 et x_2 sont sélectionnés aléatoirement.

La sécurité de notre construction repose sur celle des schémas de signatures employés mais également sur l'hypothèse $\mathsf{EXDH}-2$. Les résultats la concernant sont formellement énoncés par le théorème suivant. Bien que les deux variantes (ROM et modèle standard) soient très similaires, l'analyse de leur sécurité diffère sur l'une des propriétés et nécessite donc d'être étudiée séparément.

Théorème 30. *1. Dans le modèle de l'oracle aléatoire, et en supposant que la fonction H est résistante aux collisions, notre système de monnaie électronique divisible est anonyme sous l'hypothèse $\mathsf{EXDH}-2$, traçable si Σ_0 est EUF-SCMA sûr et Σ_1 est EUF-CMA sûr, et satisfait la propriété de non-diffamation sous l'hypothèse DL.*

2. Dans le modèle standard, et en supposant que la fonction H est résistante aux collisions, notre système de monnaie électronique divisible est anonyme sous l'hypothèse $\mathsf{EXDH}-2$, traçable si Σ_0 est EUF-SCMA sûr et Σ_1 est EUF-CMA sûr, et satisfait la propriété de non-diffamation sous l'hypothèse $q-\mathsf{SDH}$ si Σ_{ots} est un schéma de signature à usage unique dont la sécurité est forte.

4.3.3 Preuves de Sécurité

Nous prouvons dans cette section le théorème 30 relatif à la sécurité de notre construction. La version dans le ROM ne différant de celle dans le modèle standard que sur la propriété de non-diffamation, nous décrirons des preuves communes pour l'anonymat et la traçabilité.

Preuve de la traçabilité

L'objectif de l'adversaire $\mathcal{A}$ est de produire, après q_w retraits, u traces valides de transactions $\{(2^{\ell_i}, Z_i, \Pi_i)\}_{i=1}^{u}$ telles que $\texttt{Identify}((2^{\ell_i}, Z_i, \Pi_i), (2^{\ell_j}, Z_j, \Pi_j)) = \perp$ pour tout $i \neq j$ et $\sum_{i=1}^{u} 2^{\ell_i} > q_w \cdot 2^w$. Supposons que $\mathcal{A}$ réussisse, nous distinguons les 3 cas suivants.

- Cas 1 : $\exists i$ tel que Π_i contienne une mise en gage d'une paire (g_s, h_s) qui n'a jamais été signée par la banque lors de l'exécution de l'algorithme $\texttt{BKeygen}$.
- Cas 2 : $\exists i$ tel que Π_i contienne une mise en gage d'une paire $(u_1^{\mathsf{usk}}, u_2^x)$ qui n'a jamais été signée par la banque lors d'une requête $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$.
- Cas 3 : $\forall 1 \leq i \leq u$, il existe une signature $\tau_s \in \mathsf{bpk}$ sur la paire (g_s, h_s) mise en gage dans Π_i et les paires $(u_1^{\mathsf{usk}}, u_2^x)$ impliquées dans ces traces ont toute été signées lors de requêtes $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$.

La remarque 9 permet d'éliminer le troisième cas. En effet, soient $x_1, \ldots, x_{q_w}$ les scalaires associés aux pièces retirées lors des requêtes $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$. Puisqu'un montant $\sum_{i=1}^{u} 2^{\ell_i} > q_w \cdot 2^w$ a été déposé, la banque a au moins calculé les $\sum_{i=1}^{u} 2^{\ell_i}$ éléments $z_j \leftarrow e(t_{s_i}, \widetilde{g}_{s_i \mapsto f})$ pour $f \in \mathcal{F}_n(s_i)$. Ceux-ci appartenant tous (dans le cas 3) à l'ensemble $\{G^{l_f \cdot x_i}\}_{f \in \mathcal{F}_n, 1 \leq i \leq q_w}$ qui est de taille $q_w \cdot 2^n$, il existe au moins une paire (i, j), avec $i \neq j$ telle que $z_i = z_j$. Cependant, nous avons prouvé dans la remarque précédente qu'une telle collision signifie, sauf avec une probabilité négligeable, qu'un même scalaire x a été utilisé lors de ces deux transactions. Ce dernier étant lié à une unique clé publique upk, celle-ci aurait alors été retournée par l'algorithme $\texttt{Identify}$ (car H est résistante aux collisions).

Lemme 31. *Un adversaire $\mathcal{A}$ produisant des traces satisfaisant le cas 1 avec une probabilité ϵ peut être converti en un adversaire contre la sécurité EUF-SCMA du schéma Σ_0 réussissant avec une probabilité $\epsilon/(n+1)$.*

Preuve. La réduction $\mathcal{R}$ génère normalement les paramètres publics et choisit un indice $i^* \in [0, n]$. Elle envoie alors les paires (g_s, h_s), pour tout $s \in \mathcal{S}_n$ tel que $|s| = i^*$, au challenger du jeu EUF-SCMA qui lui retourne les signatures τ_s correspondantes ainsi qu'une clé publique pk. $\mathcal{R}$ peut alors

générer les autres paires $(sk_0^{(i)}, \mathsf{pk}_0^{(i)})$ pour $i \neq i^*$ et les utilise pour signer les autres sommets. Finalement, il définit $\mathsf{pk}_0^{(i^*)} = \mathsf{pk}$ et publie $\{\mathsf{pk}_0^{(i)}\}$ ainsi que les signatures τ_s. Puisque la clé $\mathsf{sk}_0^{(i)}$ n'est plus impliquée dans le reste du protocole, $\mathcal{R}$ est capable de gérer toutes les requêtes émises par $\mathcal{A}$. À la fin, celui-ci retourne u traces dont l'une d'entre elles contient une mise en gage vers une paire (g_s, h_s) que la banque n'a pas signée. La validité du système de preuve utilisé assure alors que les traces contiennent également une mise en gage d'un élément τ_s tel que $\Sigma_0.\mathtt{Verify}((g_s, h_s), \tau_s, \mathsf{pk}_0^{(n-\ell)}) = 1$. Si $\ell \neq i^*$, alors $\mathcal{R}$ arrête. Sinon, τ_s constitue une contrefaçon valide sur (g_s, h_s) pour la clé pk et peut donc être utilisé pour casser la sécurité du schéma Σ_0.

Lemme 32. *Un adversaire $\mathcal{A}$ produisant des traces satisfaisant le cas 2 avec une probabilité ϵ peut être converti en un adversaire contre la sécurité EUF-CMA du schéma Σ_1 réussissant avec la même probabilité.*

Preuve. La réduction $\mathcal{R}$ génère normalement les paramètres publics et la clé de la banque à l'exception de pk_1 qui est définie comme étant pk, la clé reçue du challenger $\mathcal{C}$ du jeu définissant la sécurité EUF-CMA. $\mathcal{R}$ est alors capable de répondre directement à toutes les requêtes de sécurité à l'exception de celles de type $\mathcal{O}\mathtt{Withdraw}_{\mathcal{B}}$ pour lesquelles il transmet les paires $(u_1^{\mathsf{usk}}, u_2^x)$ à $\mathcal{C}$ afin d'obtenir les signatures σ correspondantes.

A la fin du jeu, $\mathcal{A}$ retourne u traces de transactions dont l'une entre elles contient une mise en gage d'une paire $(u_1^{\mathsf{usk}}, u_2^x)$ jamais signée par la banque. Là encore, la validité du système de preuve implique la mise en gage de σ tel que $\Sigma_1.\mathtt{Verify}((u_1^{\mathsf{usk}}, u_2^x), \sigma, \mathsf{pk}_1) = 1$. Cet élément peut donc être extrait et utilisé pour casser la sécurité EUF-CMA de Σ_1.

Preuve de la propriété de non-diffamation

ROM. Soit $\mathcal{A}$ un adversaire contre cette propriété. Nous construisons une réduction $\mathcal{R}$ utilisant $\mathcal{A}$ contre des challenges DL dans $\mathbb{G}_1$. Soit (g, g^α) un challenge DL, $\mathcal{R}$ génère normalement les paramètres publics et choisit un indice $i^* \in [1, q_a]$ où q_a est une borne sur le nombre de requêtes $\mathcal{O}\mathtt{Add}$. $\mathcal{R}$ va alors répondre aux différentes requêtes comme suit.

- $\mathcal{O}\mathtt{Add}()$: lorsque $\mathcal{A}$ émet sa i-ème requête pour enregistrer un utilisateur, $\mathcal{R}$ exécute l'algorithme $\mathtt{Keygen}$ si $i \neq i^*$ et définit $\mathsf{upk}^* \leftarrow g^\alpha$ sinon.
- $\mathcal{O}\mathtt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ retourne la clé secrète correspondante si $\mathsf{upk} \neq \mathsf{upk}^*$. Sinon, elle arrête.
- $\mathcal{O}\mathtt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ conserve la clé publique qui est maintenant considérée comme enregistrée.
- $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}(\mathsf{bsk}, \mathsf{upk})$: $\mathcal{R}$ agit normalement si $\mathsf{upk} \neq \mathsf{upk}^*$ et simule la preuve interactive de connaissance de α dans le cas contraire.
- $\mathcal{O}\mathtt{Spend}(\mathsf{upk}, 2^\ell)$: $\mathcal{R}$ agit normalement si $\mathsf{upk} \neq \mathsf{upk}^*$ et simule la preuve non-interactive de connaissance de α dans le cas contraire.

Un adversaire remportant le jeu produit des traces $(2^{\ell_1}, Z_1, \Pi_1)$ et $(2^{\ell_2}, Z_2, \Pi_2)$ accusant l'utilisateur upk de double-dépense. Si $\mathsf{upk} \neq \mathsf{upk}^*$, alors $\mathcal{R}$ arrête. Sinon, au moins une de ces traces n'a pas été produite par $\mathcal{R}$ (sinon il aurait commis une double-dépense sur ses propres pièces). Il est alors possible d'extraire α de la signature de connaissance contenue dans la trace falsifiée. $\mathcal{R}$ est donc capable de résoudre le problème du logarithme discret vu qu'il continuera le jeu jusqu'à la fin avec une probabilité de $1/q_a$.

Modèle Standard. Un adversaire $\mathcal{A}$ remportant le jeu est capable de produire deux traces accusant un utilisateur honnête upk de double-dépense. Comme nous venons de l'expliquer dans le cas du ROM, cela signifie qu'au moins l'une d'entre elles n'a pas été produite par $\mathcal{R}$. Soit pk'_{ots} la clé publique du schéma de signature à usage unique utilisée dans la trace falsifiée, nous distinguons les deux cas suivants.

– Cas 1 : pk'_{ots} est l'une des clés que $\mathcal{R}$ a utilisées pour répondre aux requêtes $\mathcal{O}\texttt{Spend}$.
– Cas 2 : pk'_{ots} n'a jamais été utilisée par $\mathcal{R}$.

Si $\mathcal{A}$ produit des traces rentrant dans le cas 1, alors il peut être converti en un adversaire contre la sécurité de Σ_{ots}. En effet, ce schéma de signature est utilisé pour signer l'intégralité de la trace. Par conséquent, une trace falsifiée nécessitera une nouvelle signature η qui sera donc une contrefaçon pour Σ_{ots}.

Lemme 33. *Soit q_s (resp. q_a) une borne sur le nombre de requêtes $\mathcal{O}\texttt{Spend}$ (resp. $\mathcal{O}\texttt{Add}$). Un adversaire produisant des traces satisfaisant le cas 2 avec probabilité ϵ peut être transformé en un adversaire contre l'hypothèse $q_s - \mathsf{SDH}$ réussissant avec probabilité ϵ/q_a.*

Preuve. Soit $(g, g^{\alpha}, \ldots, g^{\alpha_{q_s}})$ un challenge $q_s - \mathsf{SDH}$. $\mathcal{R}$ choisit un indice $i^* \in [1, q_a]$ et génère normalement les paramètres publics sauf u_1 qu'il définit comme étant g^z pour un certain scalaire z. De plus, elle calcule q_s paires de clés $(\mathsf{sk}^{(i)}_{ots}, \mathsf{pk}^{(i)}_{ots}) \leftarrow \Sigma_{ots}.\texttt{Keygen}(1^k)$ et définit $w \leftarrow g^{\prod_{i=1}^{q_s}(\alpha + H(\mathsf{pk}^{(i)}_{ots}))}$ (ce qui peut être calculé à partir du challenge $q_s - \mathsf{SDH}$). $\mathcal{R}$ peut alors répondre aux différentes requêtes de la manière suivante.

– $\mathcal{O}\texttt{Add}()$: lorsque $\mathcal{A}$ émet la i-ème requête $\mathcal{O}\texttt{Add}$, $\mathcal{R}$ exécute l'algorithme $\texttt{Keygen}$ si $i \neq i^*$ et définit $\mathsf{upk}^* \leftarrow g^{\alpha}$ dans le cas contraire.
– $\mathcal{O}\texttt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ retourne la clé secrète si $\mathsf{upk} \neq \mathsf{upk}^*$. Sinon, elle arrête.
– $\mathcal{O}\texttt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ conserve la clé publique upk qui est maintenant considérée comme enregistrée.
– $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}(\mathsf{bsk}, \mathsf{upk})$: $\mathcal{R}$ agit normalement si $\mathsf{upk} \neq \mathsf{upk}^*$ et simule la preuve interactive de connaissance de α dans le cas contraire.
– $\mathcal{O}\texttt{Spend}(\mathsf{upk}, 2^{\ell})$: $\mathcal{R}$ agit normalement si $\mathsf{upk} \neq \mathsf{upk}^*$. Sinon, pour répondre à la $j-$ème requête sur upk^*, elle commence par sélectionner la paire $(sk^{(j)}_{ots}, pk^{(j)}_{ots})$ et calcule l'élément $\mu \leftarrow g^{\prod_{i=1, i\neq j}^{q_s}(\alpha + H(pk^{(i)}_{ots}))}$ qui vérifie $\mu = w^{\frac{1}{\alpha + H(pk^{(j)}_{ots})}}$. Il peut alors utiliser $sk^{(j)}_{ots}$ comme spécifié dans le protocole $\texttt{Spend}$.

$\mathcal{A}$ finit par retourner deux traces valides $(2^{\ell_1}, Z_1, \Pi_1)$ et $(2^{\ell_2}, Z_2, \Pi_2)$ accusant upk de double-dépense. Si $\mathsf{upk} \neq \mathsf{upk}^*$, alors $\mathcal{R}$ arrête. La validité du système de preuve utilisé implique que la trace falsifiée a été signée en utilisant une clé sk_{ots} et donc qu'elle contient une mise en gage de $\mu = w^{\frac{1}{\alpha + H(\mathsf{pk}_{ots})}}$. Puisque nous sommes dans le cas 2, $pk_{ots} \notin \{pk^{(i)}_{ots}\}_i$, et donc $H(pk_{ots}) \notin \{H(pk^{(i)}_{ots})\}_i$ car la fonction H est résistante aux collisions. L'élément μ peut alors être utilisé pour résoudre le problème q_s-SDH dans $\mathbb{G}_1$ (comme expliqué dans [BB08]). $\mathcal{R}$ est donc capable de résoudre ce problème puisque le jeu ne sera pas arrêté avec probabilité $1/q_a$.

Preuve de l'anonymat

Soit $\mathcal{A}$ un adversaire contre l'anonymat dont l'avantage est ϵ. Nous construisons une réduction $\mathcal{R}$ utilisant $\mathcal{A}$ contre le problème $\mathsf{EXDH} - 2$. Soient $(g, h, g^x, h^x, g^a, h^a, g^{y\cdot a}, h^{y\cdot a}, g^{z_1}, h^{z_2}) \in \mathbb{G}_1^{10}$ et $(\widetilde{g}, \widetilde{g}^a, \widetilde{g}^y) \in \mathbb{G}_2^3$ un challenge $\mathsf{EXDH} - 2$. $\mathcal{R}$ choisit un sommet $s^* \in \mathcal{S}_n$ et va générer les paramètres du système de la manière suivante.

– $(u_1, u_2) \leftarrow (g^{d_1}, g^{d_2})$ pour des scalaires $d_1, d_2 \xleftarrow{\$} \mathbb{Z}_p$.
– Pour tout $f \in \mathcal{F}_n$, $l_f \xleftarrow{\$} \mathbb{Z}_p$.
– Pour tout $s \in \mathcal{S}_n$, $r_s \xleftarrow{\$} \mathbb{Z}_p$.
– $g_{s^*} \leftarrow (g^{y\cdot a})^{r_{s^*}}$.
– $\widetilde{g}_{s^* \mapsto f} \leftarrow \widetilde{g}^{\frac{l_f}{r_{s^*}}}$ pour tout $f \in \mathcal{F}_n(s^*)$.
– Pour tout $s \in \mathcal{S}_n$:
  – si s est un suffixe de s^*, alors $(g_s, h_s) \leftarrow ((g^{y\cdot a})^{r_s}, (h^{y\cdot a})^{r_s})$;
  – si s^* est un préfixe de s, alors $(g_s, h_s) \leftarrow ((g^a)^{r_s}, (h^a)^{r_s}$;

  - sinon, $(g_s, h_s) \leftarrow (g^{r_s}, h^{r_s})$.
- Pour tout $s \in \mathcal{S}_n$ et tout $f \in \mathcal{F}_n$:
  - si s^* est un suffixe de s, alors $\widetilde{g}_{s\mapsto f} \leftarrow \widetilde{g}^{\frac{l_f}{r_s}}$;
  - si s est un préfixe de s^* et $f \in \mathcal{F}_n(s^*)$, alors $\widetilde{g}_{s\mapsto f} \leftarrow (\widetilde{g}^{y})^{\frac{l_f}{r_s}}$;
  - si s est un préfixe de s^* et $f \notin \mathcal{F}_n(s^*)$, alors $\widetilde{g}_{s\mapsto f} \leftarrow \widetilde{g}^{\frac{l_f}{r_s}}$;
  - sinon, $\widetilde{g}_{s\mapsto f} \leftarrow (\widetilde{g}^{a})^{\frac{l_f}{r_s}}$.

Ainsi générés, les paramètres publics sont valides puisque, pour tout $f \in \mathcal{F}_n$, on a :
- $e(g_s, \widetilde{g}_{s\mapsto f}) = e(g, \widetilde{g})^{a\cdot y\cdot l_f}$ si $f \in \mathcal{F}_n(s^*)$;
- $e(g_s, \widetilde{g}_{s\mapsto f}) = e(g, \widetilde{g})^{a\cdot l_f}$ sinon.

Dans le modèle standard, $\mathcal{R}$ génère également une chaîne de référence commune vérifiant la propriété d'indistinguabilité des témoins. Soit q_w une borne sur le nombre de requêtes $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}$, $\mathcal{R}$ choisit un indice $i^* \in [1, q_w]$ et répond aux différentes requêtes comme suit.
- $\mathcal{O}\texttt{Add}()$: $\mathcal{R}$ exécute l'algorithme `Keygen` et retourne `upk` (ou `mpk`).
- $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}(\mathsf{bsk}, \mathsf{upk})$: lors de la $i-$ème requête à l'oracle $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}$, $\mathcal{R}$ agit normalement si $i \neq i^*$ et comme si la valeur secrète de la pièce était x autrement (en envoyant $(g^x)^{d_2}$ et en simulant la preuve de connaissance de x). La clé publique impliquée dans ce retrait sera alors notée upk^*.
- $\mathcal{O}\texttt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ agit normalement si la requête ne porte pas sur upk^*. Sinon, elle arrête le jeu.
- $\mathcal{O}\texttt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ conserve la clé publique `upk` qui est maintenant considérée comme enregistrée.
- $\mathcal{O}\texttt{Spend}(\mathsf{upk}, 2^\ell)$: $\mathcal{R}$ est capable de gérer toutes les requêtes pour lesquelles $\mathsf{upk} \neq \mathsf{upk}^*$ puisqu'elle connait tous les secrets de ces utilisateurs. Si la requête porte sur upk^*, la réduction reste capable de répondre tant que $c_{\mathsf{upk}^*} \leq m_{\mathsf{upk}^*} \cdot 2^n - 2^\ell - 2^{n-|s^*|}$. En effet, cette condition implique qu'il existe au moins un sommet s non dépensé qui n'est ni un préfixe, ni un suffixe de s^*. Par conséquent, la réduction peut calculer une paire valide $(t_s, v_s) \leftarrow ((g^x)^{r_s}, \mathsf{upk}^R \cdot (h^x)^{r_s})$ où $R \leftarrow H(info)$ et simuler la preuve non-interactive de connaissance (ce qui est possible, même dans le modèle standard, grâce au type de CRS générée par $\mathcal{R}$). Dans le cas où la condition sur c_{upk^*} n'est pas vérifiée, $\mathcal{R}$ arrête la simulation.

Lors de la phase dite *de challenge*, $\mathcal{A}$ choisit deux identités upk_0 et upk_1 ainsi qu'une valeur 2^ℓ. Il est bien évidemment exigé que chacun de ces utilisateurs dispose encore d'une valeur d'au moins 2^l, ce qui se traduit par la condition $c_{upk_b} \leq m_{\mathsf{upk}_b} \cdot 2^n - 2^\ell$. Si $\mathsf{upk}^* \notin \{\mathsf{upk}_0, \mathsf{upk}_1\}$ ou si $\ell > n - |s^*|$, alors $\mathcal{R}$ arrête. Sinon, il choisit un sommet s de longueur $n - \ell$ dont s^* est un préfixe (ou s^* directement si $|s^*| = n - \ell$) qui ne peut pas avoir été dépensé en raison de la restriction imposée aux requêtes $\mathcal{O}\texttt{Spend}$. Comme $(g_s, h_s) = ((g^{y\cdot a})^{r_s}, (h^{y\cdot a})^{r_s})$, il faut retourner $((g^{x\cdot y\cdot a})^{r_s}, (\mathsf{upk}^*)^R \cdot (h^{x\cdot y\cdot a})^{r_s})$, ce qui correspond exactement à $((g^{z_1})^{r_s}, (\mathsf{upk}^*)^r \cdot (h^{z_2})^{r_s})$ si $z_1 = x \cdot y \cdot a = z_2$. $\mathcal{R}$ peut alors simuler la preuve de connaissance et répondre aux différentes requêtes comme précédemment. La simulation est alors parfaite. Dans le cas où z_1 et z_2 sont aléatoires, ce qui est indistinguable sous l'hypothèse $\mathsf{EXDH} - 2$, l'identité upk^* est alors parfaitement masquée.

Cependant, pour que $\mathcal{R}$ poursuive la simulation jusqu'à la fin, il est nécessaire que certaines conditions soient satisfaites. Tout d'abord, il est nécessaire que la pièce créée lors du i^*-ème retrait appartienne à l'un des deux utilisateurs impliqués dans la phase de challenge, ce qui arrive avec une probabilité supérieure à $\frac{2}{q_w}$. Ensuite, il faut également que le choix du niveau de s^* soit bon, c'est-à-dire que $|s^*| = n - \ell$, ce qui arrive avec probabilité $\frac{1}{n+1}$. En effet, si $\ell < n - |s^*|$, alors $\mathcal{R}$ peut arrêter la simulation lors des requêtes $\mathcal{O}\texttt{Spend}$, et si $\ell > n - |s^*|$ alors $\mathcal{R}$ arrêtera lors de la phase de challenge car elle sera incapable d'introduire correctement les éléments du challenge $\mathsf{EXDH} - 2$. Ainsi $\mathcal{R}$ continuera la simulation avec une probabilité supérieure à $\frac{2}{q_w \cdot (n+1)}$.

Remarque 10. On peut remarquer que les éléments $h, h^x, h^a, h^{y \cdot a}$ et h_2^z ne servent qu'à simuler les éléments v_s fournis lors d'une dépense. Ces derniers n'étant utiles que pour satisfaire la propriété d'anonymat, il est plus simple d'utiliser l'hypothèse $\mathsf{EXDH}-1$ dans les cas où un niveau inférieur d'anonymat (par exemple, la non-corrélation ou sa variante forte) serait suffisant.

4.4 Un Système Déployable à Grande Échelle

Nous présentons dans cette section une amélioration du schéma précédent présentée dans l'article *Scalable Divisible E-Cash* [CPST15b] publié à la conférence ACNS 2015.

4.4.1 Idée Générale

Comme nous venons de le voir, la construction précédente bénéfice d'une efficacité remarquable lors des retraits ou des dépenses mais souffre de deux défauts, à savoir la taille de ses paramètres publics et le coût pour la banque de la procédure de dépôt. Si le premier problème ne semble pas insurmontable (nous donnons une estimation de la taille des paramètres publics dans la prochaine section), le deuxième est nettement plus ennuyeux pour un déploiement à grande échelle d'un système de monnaie électronique. En effet, devoir calculer, puis conserver, 2^n numéros de série pour chaque dépôt est problématique pour un système gérant des millions de transactions.

Ces deux problèmes sont en réalité liés à la construction de l'arbre. En effet, rappelons que chaque sommet est associé à un élément $g_s \leftarrow g^{r_s}$ pour un certain scalaire $r_s \xleftarrow{\$} \mathbb{Z}_p$. Le fait que les éléments g_s soient indépendants les uns des autres a comme conséquence que chaque sommet s doit disposer de son propre ensemble $\{\widetilde{g}_{s \mapsto f}\}_{f \in \mathcal{F}_n(s)}$. Cela entraîne d'une part l'augmentation de la taille des paramètres publics mais également le problème rencontré lors du dépôt. En effet, un schéma fortement non-corrélable ou anonyme ne révèle pas le sommet s dépensé, ce qui ne laisse pas d'autre choix à la banque que de calculer, pour chaque sommet s' de niveau $|s|$, les éléments $z_i \leftarrow e(t_s, \widetilde{g}_{s' \mapsto f})$ pour tout $f \in \mathcal{F}_n(s')$.

Pour éviter ces calculs inutiles, sans pour autant révéler le sommet s, une solution serait de rendre les éléments nécessaires au calcul des numéros de série (à savoir les $\widetilde{g}_{s \mapsto f}$) commun à tous sommets d'un même niveau. La banque n'aurait ainsi plus besoin de reproduire la procédure pour chaque ensemble $\{\widetilde{g}_{s' \mapsto f}\}_{f \in \mathcal{F}_n(s')}$ avec $|s'| = |s|$ mais une seule fois pour l'ensemble $\{\widetilde{g}_{|s|,f}\}_{f \in \mathcal{L}(|s|)}$. Par ailleurs, cela réduirait significativement la taille des paramètres publics. Cependant, cela nécessiterait certaines relations entre les éléments d'un même niveau, ce qui n'est pas le cas avec la construction précédente. Nous proposons ici une nouvelle manière de construire la structure d'arbre commune qui permet de résoudre ces problèmes. Celle-ci impose plusieurs modifications par rapport au schéma précédent et entraîne donc la définition d'un nouveau système de monnaie électronique divisible.

Description. L'idée de cette nouvelle construction est d'associer la racine ϵ de l'arbre à un élément aléatoire $g_\epsilon \leftarrow g^{y_\epsilon}$ et chaque niveau i ($i \in [0, n]$) à deux scalaires aléatoires $y_{i,0}$ et $y_{i,1}$. L'arbre peut alors être défini récursivement de la manière suivante. Soit $s \in \mathbb{G}_1$ l'élément associé à un sommet s, l'élément associé à son fils gauche est $g_{s||0} \leftarrow g_s^{y_{|s|+1,0}}$ et celui associé à son fils droit est $g_{s||1} \leftarrow g_s^{y_{|s|+1,1}}$. Par conséquent, comme l'illustre la figure 4.5, chaque sommet $s = b_1, \ldots, b_{|s|}$ est associé à l'élément $g_s \leftarrow g^{y_\epsilon \prod_{i=1}^{|s|} y_{i,b_i}}$.

Pour permettre à la banque de calculer les numéros de série associés à chacun de ces sommets, nous allons fournir, pour chaque niveau i et chaque $f \in \mathcal{L}(i)$, l'élément $\widetilde{g}_{i,f} \leftarrow \widetilde{g}^{\prod_{j=i+1}^{n} y_{j,b_j}}$. Cela définit implicitement l'ensemble des numéros de série associés à une pièce dont le secret est x comme étant $\{G^{x \cdot y_\epsilon \prod_{i=1}^{n} y_{i,b_i}}\}_{b_1 \ldots b_n \in \mathcal{F}_n}$ (*cf* figure 4.6 pour une description générique, sans le secret x). Le point essentiel ici est que ces éléments sont communs à tous les sommets d'un même niveau i et seront donc utilisés à chaque fois qu'un dépôt d'un montant de 2^{n-i} est réalisé.

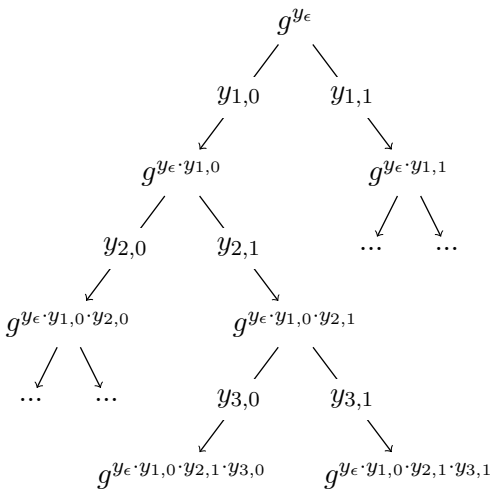


Figure 4.5 – Pièce Divisible

Malheureusement, il n'est alors plus possible de fournir, lors d'une dépense, l'élément $t_s \leftarrow g_s^x$ sans mettre en danger l'anonymat du schéma. En effet, considérons par exemple un sommet s de niveau $n-1$. Une dépense impliquant son fils gauche $s||0$ ne devrait pas pouvoir être relié à une dépense impliquant son fils droit $s||1$. Cela n'est pas vrai si nous révélons $g_{s||0}^x$ et $g_{s||1}^x$ puisqu'il est possible de vérifier si l'égalité $e(g_{s||0}^x, \widetilde{g}_{n-1,1}) = e(g_{s||1}^x, \widetilde{g}_{n-1,0})$ tient ou non. En effet :

$$e(g_{s||0}^x, \widetilde{g}_{n-1,1}) = e(g_s^x, \widetilde{g})^{y_{n,0} \cdot y_{n,1}} = e(g_{s||1}^x, \widetilde{g}_{n-1,0})$$

Intuitivement, le problème vient du fait qu'il est possible de combiner, au travers du couplage, l'élément t_s avec tous les éléments $\widetilde{g}_{i,f}$, quel que soit le niveau i leur étant associé. Si nous reprenons notre exemple, nous remarquons que $t_{s||0} = t_s^{y_{n,0}}$ et que $t_{s||1} = t_s^{y_{n,1}}$. Cette différence d'exposants peut être compensée à l'aide du couplage en utilisant les éléments $\widetilde{g}_{n-1,0} = \widetilde{g}^{y_{n,0}}$ et $\widetilde{g}_{n-1,1} = \widetilde{g}^{y_{n,1}}$ prévus initialement pour calculer les numéros de série à partir de sommets de niveau $n-1$. Plus généralement, le fait que les scalaires $y_{j,b}$ (pour $j \leq |s|$) intervenant dans la construction de g_s (et donc de t_s) soient également impliqués dans la construction des éléments $\widetilde{g}_{i,f}$ pour $i < |s|$ rend ce type de relations possible. Il est donc indispensable, pour préserver l'anonymat, de rendre impossible la combinaison d'éléments t_s avec des éléments $\widetilde{g}_{i,f}$ de niveau $i \neq |s|$.

Pour ce faire, l'utilisateur va désormais envoyer, lors d'une dépense, un chiffré El Gamal de $t_s = g_s^x$ sous une certaine clé publique $k_{|s|}$, c'est-à-dire une paire $(g^{r_1}, t_s \cdot k_{|s|}^{r_1})$ pour un scalaire aléatoire r_1. Il va procéder de même pour l'élément $v_s = \mathsf{upk}^R \cdot h_s^x$ en envoyant une paire $(g^{r_2}, v_s \cdot k_{|s|}^{r_2})$ avec $r_2 \xleftarrow{\$} \mathbb{Z}_p$. La banque devant pouvoir retrouver les numéros de série déterministes, il est nécessaire de fournir des éléments $\widetilde{h}_{|s|,f}$ (dont la construction est précisée ci-dessous) permettant de supprimer l'aléa rajouté. Cette approche peut surprendre car d'un côté elle impose à l'utilisateur le chiffrement de certains éléments et de l'autre elle fournit un moyen d'annuler celui-ci. Cependant, elle permet, comme nous allons le voir, d'empêcher l'utilisation des éléments $\widetilde{g}_{i,f}$ pour relier des dépenses entres elles.

4.4.2 Construction

Nous décrivons à présent formellement les différents algorithmes constituant notre nouveau système de monnaie électronique. Tout comme le précédent, notre schéma peut être décliné dans le ROM ou dans le modèle standard. Cependant, pour plus de simplicité, nous choisissons ici de ne décrire que le cas du modèle standard.

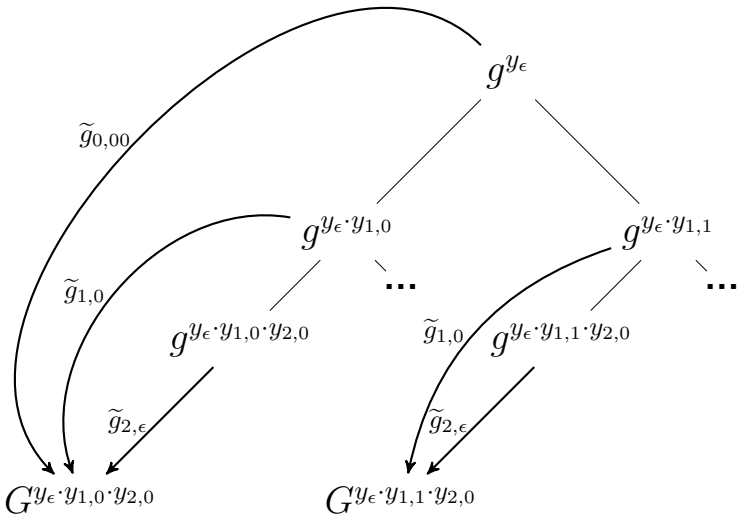


Figure 4.6 – Calcul des numéros de série

– `Setup`$(1^k, V)$: soient $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires d'ordre p, des générateurs g, h, u_1, u_2, w de $\mathbb{G}_1$ et $\widetilde{g}$ un générateur de $\mathbb{G}_2$. L'entité générant les paramètres choisit $(y_\epsilon, a_0) \xleftarrow{\$} \mathbb{Z}_p^2$ et, pour tout $i = 1, \ldots, n$, le triplet $(y_{i,0}, y_{i,1}, a_i) \xleftarrow{\$} \mathbb{Z}_p^3$. Elle calcule alors $(g_\epsilon, h_\epsilon) \leftarrow (g^{y_\epsilon}, h^{y_\epsilon})$ et $(g_s, h_s) \leftarrow (g^{y_\epsilon \prod_{i=1}^{|s|} y_{i,b_i}}, h^{y_\epsilon \prod_{i=1}^{|s|} y_{i,b_i}})$ pour chaque sommet $s = b_1 \ldots b_{|s|}$. Finalement, elle calcule pour tout $i = 0, \ldots, n$:
  – $k_i \leftarrow g^{a_i}$, à savoir une clé de chiffrement El Gamal ;
  – $(\widetilde{g}_{i,f}, \widetilde{h}_{i,f}) \leftarrow (\widetilde{g}^{\prod_{j=i+1}^{n} y_{j,b_j}}, \widetilde{g}^{-a_i \prod_{j=i+1}^{n} y_{j,b_j}})$, pour tout $f = b_{i+1} \ldots b_n \in \mathcal{L}(i)$.

  Comme nous l'avons expliqué dans la section précédente, il peut être préférable de générer ces paramètres coopérativement par la banque et un ensemble d'utilisateurs.
  Les paramètres publics *p.p.* contiennent la description $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ des groupes bilinéaires, les générateurs g, h, u_1, u_2, w et $\widetilde{g}$, une fonction de hachage $H : \{0,1\}^* \to \mathbb{Z}_p$ résistante aux collisions, ainsi que les éléments $\{(g_s, h_s)\}_{s \in \mathcal{S}_n}$, $\{k_i\}_{i=0,\ldots,n}$ et $\{(\widetilde{g}_{i,f}, \widetilde{h}_{i,f})\}_{i=0,\ldots,n, f \in \mathcal{L}(i)}$.
  On peut remarquer que les paires $(\widetilde{g}_{i,f}, \widetilde{h}_{i,f})$ peuvent être utilisées pour annuler le chiffrement El Gamal au niveau i, pour toute feuille f :

$$e(k_i, \widetilde{g}_{i,f}) = e(g^{a_i}, \widetilde{g}^{\prod_{j=i+1}^{n} y_{j,b_j}}) = (g, \widetilde{g}^{a_i \prod_{j=i+1}^{n} y_{j,b_j}}) = e(g, \widetilde{h}_{i,f})^{-1}.$$

  Par conséquent, le chiffrement sous les clés k_i ne sera évidemment pas sémantiquement sûr mais sera suffisant pour atteindre les propriétés de sécurité souhaitées.
  Pour finir, les paramètres publics contiennent également la description d'une *CRS* pour le système de preuve Groth-Sahai ainsi que celle d'un schéma de signature Σ_{ots} à usage unique (par exemple celui de [BB08]).
– `BKeygen`() : la banque, devant signer deux types de messages, sélectionne deux schémas Σ_0 and Σ_1.
  – Le premier sera utilisé pour calculer des signatures τ_s sur les paires (g_s, h_s) pour tout sommet s de l'arbre. Ces signatures permettront à l'utilisateur de prouver, lors d'une dépense, qu'il a utilisé une paire (g_s, h_s) valide sans la révéler.
  – Le second sera utilisé par la banque lors du protocole `Withdraw` pour certifier les valeurs secrètes associées à la pièce retirée.

Les deux schémas doivent permettre de signer des éléments de $\mathbb{G}_1^2$ tout en étant compatibles avec les preuves Groth-Sahai. Nous choisirons de les implémenter à l'aide de la construction proposée dans [AGHO11] puisqu'elle a été prouvée optimale pour les couplages de type 3.
La banque génère les paires de clés $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \Sigma_1.\mathtt{Keygen}(p.p.)$ ainsi que $(\mathsf{sk}_0^{(i)}, \mathsf{pk}_0^{(i)}) \leftarrow \Sigma_0.\mathtt{Keygen}(p.p.)$, pour chaque niveau $i = 0, \ldots, n$ de l'arbre, et calcule, pour chaque sommet $s \in \mathcal{S}_n$, $\tau_s \leftarrow \Sigma_0.\mathtt{Sign}(\mathsf{sk}_0^{(|s|)}, (g_s, h_s))$. Finalement, elle définit bsk comme étant sk_1 et publie $\mathsf{bpk} \leftarrow (\{\mathsf{pk}_0^{(i)}\}_i, \mathsf{pk}_1, \{\tau_s\}_{s\in\mathcal{S}_n})$. La remarque ci-dessous décrit un moyen de réduire la taille de cette clé publique.

– $\mathtt{Keygen}()$: Chaque utilisateur (resp. marchand) choisit un scalaire aléatoire $\mathsf{usk} \leftarrow \mathbb{Z}_p$ (resp. msk) et obtient $\mathsf{upk} \leftarrow g^{\mathsf{usk}}$ (resp. $\mathsf{mpk} \leftarrow g^{\mathsf{msk}}$). Dans ce qui suit, nous supposons que la clé upk (resp. mpk) est publique, c'est-à-dire que tout le monde peut en obtenir une copie authentique.

– $\mathtt{Withdraw}(\mathcal{B}(\mathsf{bsk}, \mathsf{upk}), \mathcal{U}(\mathsf{usk}, \mathsf{bpk}))$: Pour retirer une pièce divisible, l'utilisateur doit calculer conjointement avec la banque un scalaire aléatoire x, puis obtenir un certificat dessus. En pratique, x peut être calculé comme étant $x_1 + x_2$ où x_1 est choisi et gardé secret par l'utilisateur tandis que x_2 est choisi par la banque qui le lui transmet. Il est également nécessaire de lier cette valeur secrète x à l'identité de l'utilisateur pour pouvoir identifier les auteurs de fraudes.
Pour cela, l'utilisateur, calcule u_1^{usk} et $u_2^{x_1}$ et les envoie, ainsi que upk, à la banque à qui il prouve sa connaissance de x_1 et usk (en utilisant, par exemple, le protocole interactif de Schnorr). Si la preuve est valide, la banque choisit un élément x_2 aléatoire et vérifie que $u = u_2^{x_1} \cdot u_2^{x_2}$ n'a pas déjà été utilisé, calcule $\sigma \leftarrow \Sigma_1.\mathtt{Sign}(\mathsf{sk}_1, (u_1^{\mathsf{usk}}, u))$ et l'envoie à l'utilisateur, en même temps que x_2. L'utilisateur reconstruit alors $x = x_1 + x_2$, et conserve $C \leftarrow (x, \sigma)$. La signature σ portant sur $(u_1^{\mathsf{usk}}, u_2^x)$ garantit le lien entre l'utilisateur et le secret x.

– $\mathtt{Spend}(\mathcal{U}(\mathsf{usk}, C, \mathsf{bpk}, \mathsf{mpk}, 2^\ell), \mathcal{M}(\mathsf{msk}, \mathsf{bpk}, 2^\ell))$: Pour dépenser une valeur 2^ℓ, l'utilisateur choisit un sommet s non dépensé de niveau $n-\ell$ ainsi que deux scalaires aléatoires $r_1, r_2 \leftarrow \mathbb{Z}_p$ et calcule $R \leftarrow H(info)$, $t_s \leftarrow (g^{r_1}, g_s^x \cdot k_{n-\ell}^{r_1})$ et $v_s \leftarrow (g^{r_2}, \mathsf{upk}^R \cdot h_s^x \cdot k_{n-\ell}^{r_2})$, où $info$ regroupe des informations relatives à la transaction (telles que la date, le montant, la clé publique du marchand, ...). Il doit ensuite prouver que t_s et v_s sont valides, c'est-à-dire qu'ils ont été calculés à l'aide d'éléments certifiés lors d'un retrait et d'une paire (g_s, h_s) figurant dans les paramètres publics. Pour ce faire, il va fournir une preuve Groth-Sahai de connaissance de σ ainsi que d'une signature τ_s portant sur (g_s, h_s).
L'utilisateur commence par générer une paire de clés $(\mathsf{sk}_{ots}, \mathsf{pk}_{ots}) \leftarrow \Sigma_{ots}.\mathtt{Keygen}(1^k)$ ainsi que $\mu \leftarrow w^{\frac{1}{\mathsf{usk}+H(\mathsf{pk}_{ots})}}$. Il calcule ensuite les mises en gages de $\mathsf{usk}, x, r_1, r_2, g_s, h_s, \tau_s, \sigma, \mu$, $U_1 = u_1^{\mathsf{usk}}$, et $U_2 = u_2^x$. Il peut alors générer une preuve non-interactive à divulgation nulle π que celles-ci vérifient :

$$t_s = (g^{r_1}, g_s^x \cdot k_{n-\ell}^{r_1}) \quad \wedge \quad v_s = (g^{r_2}, (g^R)^{\mathsf{usk}} \cdot h_s^x \cdot k_{n-\ell}^{r_2})$$

$$U_2 = u_2^x \quad \wedge \quad U_1 = u_1^{\mathsf{usk}} \quad \wedge \quad \mu^{(\mathsf{usk}+H(\mathsf{pk}_{ots}))} = w$$

ainsi qu'une preuve non-interactive satisfaisant l'indistinguabilité des témoins que les signatures mises en gages sont valides :

$$1 = \Sigma_0.\mathtt{Verify}(\mathsf{pk}_0^{(n-\ell)}, (g_s, h_s), \tau_s) \quad \wedge \quad 1 = \Sigma_1.\mathtt{Verify}(\mathsf{pk}_1, (U_1, U_2), \sigma).$$

Il ne reste alors plus qu'à signer l'intégralité des données de la transaction en calculant $\eta \leftarrow \Sigma_{ots}.\mathtt{Sign}(\mathsf{sk}_{ots}, H(R||t_s||v_s||\pi||\pi'))$ qui est envoyé à $\mathcal{M}$ en même temps que $\mathsf{pk}_{ots}, t_s, v_s, \pi, \pi'$. Ce dernier peut alors vérifier la validité des preuves et de la signature et accepte si tout est correct auquel cas il conserve $(2^\ell, Z, \Pi)$ où $Z \leftarrow (t_s, v_s)$ et $\Pi \leftarrow (\pi, \pi', \mathsf{pk}_{ots}, \eta)$.

– $\texttt{Deposit}(\mathcal{M}(\mathsf{msk}, \mathsf{bpk}, (2^\ell, Z, \Pi)), \mathcal{B}(\mathsf{bsk}, L, \mathsf{mpk}))$: lorsqu'une trace $(2^\ell, Z = (t_s, v_s), \Pi = (\pi, \pi', \mathsf{pk}_{ots}, \eta))$ est déposée, la banque commence par en vérifier la validité et s'assure que celle-ci n'a pas déjà fait l'objet d'un dépôt. Elle sélectionne ensuite $t_s = (t_s[1], t_s[2])$ et calcule, pour tout $f \in \mathcal{L}(n-\ell)$, les numéros de série $z_f \leftarrow e(t_s[2], \widetilde{g}_{n-\ell,f}) \cdot e(t_s[1], \widetilde{h}_{n-\ell,f})$ et teste si $z_f \in L$. Si aucun ne figure dans cette liste, la banque les y rajoute et conserve la trace $(2^\ell, Z, \Pi)$. Sinon, il existe un élément $z' \in L$ tel que, pour un certain f, $z_f = z'$. La banque retrouve alors la trace $(2^{\ell'}, Z', \Pi')$ correspondante et publie $[(2^\ell, Z, \Pi), (2^{\ell'}, Z', \Pi')]$. On peut remarquer que, comme $e(k_i, \widetilde{g}_{i,f}) \cdot e(g, \widetilde{h}_{i,f}) = 1$ pour tout niveau i et toute feuille f, l'élément $z_f = e(g_s^x, \widetilde{g}_{n-\ell,f})$ est indépendant de l'aléa r_1.
Comme nous l'avons fait remarquer dans la construction de la section précédente, la banque n'a pas réellement besoin de conserver les éléments $z_i \in \mathbb{G}_T$, mais seulement leur empreinte pour une certaine fonction de hachage.

– $\texttt{Identify}((2^{\ell_1}, Z_1, \Pi_1), (2^{\ell_2}, Z_2, \Pi_2), \mathsf{bpk})$: afin d'identifier l'auteur d'une fraude, on commence par s'assurer de la validité des deux traces. Si l'une des deux n'est pas correcte, on retourne $\perp$. En posant $Z_i = (t_{s_i}, v_{s_i})$ on peut calculer les listes $S_i \leftarrow \{e(t_{s_i}[2], \widetilde{g}_{n-\ell_i,f}) \cdot e(t_{s_i}[1], \widetilde{h}_{n-\ell_i,f}), \forall f \in \mathcal{L}(n-\ell_i)\}$, pour $i = 1, 2$. Là encore, on retourne $\perp$ si l'intersection des ensembles S_1 est S_2 vide. Sinon, il existe $f_1 \in \mathcal{L}(n-\ell_1)$ et $f_2 \in \mathcal{L}(n-\ell_2)$ tels que

$$\begin{aligned} e(g_{s_1}^{x_1}, \widetilde{g}_{n-\ell_1,f_1}) &= e(t_{s_1}[2], \widetilde{g}_{n-\ell_1,f_1}) \cdot e(t_{s_1}[1], \widetilde{h}_{n-\ell_1,f_1}) \\ &= e(t_{s_2}[2], \widetilde{g}_{n-\ell_2,f_2}) \cdot e(t_{s_2}[1], \widetilde{h}_{n-\ell_2,f_2}) = e(g_{s_2}^{x_2}, \widetilde{g}_{n-\ell_2,f_2}). \end{aligned}$$

Puisque les secrets associés aux pièces sont choisis aléatoirement et de manière coopérative entre la banque et l'utilisateur et que les autres éléments impliqués dans cette égalité sont fixés depuis l'initialisation du système, une collision pour des x_i ou f_i différents est très improbable. On peut donc conclure qu'il s'agit bien d'une double-dépense sur une feuille $f \in \mathcal{F}_n$ pour un arbre paramétré par $x = x_1 = x_2$.
Par ailleurs, la signature σ et la validité des preuves garantissent que la même identité est impliquée dans les deux traces. Par conséquent, si on pose $T_i = e(v_{s_i}[2], \widetilde{g}_{n-\ell_i,f_i}) \cdot e(v_{s_i}[1], \widetilde{h}_{n-\ell_i,f_i})$, pour $i = 1, 2$, nous avons $T_i = e(\mathsf{upk}_i^{R_i} \cdot h_{s_i}^{x_i}, \widetilde{g}_{n-\ell_i,f_i})$, car $e(h_{s_1}^{x_1}, \widetilde{g}_{n-\ell_1,f_1}) = e(h_{s_2}^{x_2}, \widetilde{g}_{n-\ell_2,f_2})$, et donc :

$$T_1/T_2 = e(\mathsf{upk}^{R_1}, \widetilde{g}_{n-\ell_1,f_1})/e(\mathsf{upk}^{R_2}, \widetilde{g}_{n-\ell_2,f_2}) = e(\mathsf{upk}, \widetilde{g}_{n-\ell_1,f_1}^{R_1}/\widetilde{g}_{n-\ell_2,f_2}^{R_2}).$$

Pour identifier l'auteur de la fraude, il ne reste alors qu'à calculer, pour toutes les clés publiques upk_i, la valeur $e(\mathsf{upk}_i, \widetilde{g}_{n-\ell_1,f_1}^{R_1}/\widetilde{g}_{n-\ell_2,f_2}^{R_2})$ jusqu'à obtenir une égalité avec T_1/T_2, auquel cas l'algorithme retourne upk_i. Comme pour la construction précédente, la propriété de résistance aux collisions de H assure que l'élément $\widetilde{g}_{n-\ell_1,f_1}^{R_1}/\widetilde{g}_{n-\ell_2,f_2}^{R_2}$ ne s'annule qu'avec une probabilité négligeable.

Remarque 11. Il est indispensable, lors d'une dépense, de prouver qu'une paire (g_s, h_s) valide a été utilisée et que le niveau $|s|$ de celle-ci correspond au montant 2^ℓ dépensée (*i.e.* $|s| = n - \ell$). En effet, dans le cas contraire, les éléments z_i calculés par la banque lors d'un dépôt n'auraient plus aucun sens et ne permettraient donc plus de détecter d'éventuelles double-dépenses.

La solution que nous venons de décrire consiste à signer chaque paire (g_s, h_s) à l'aide d'une clé dépendant du niveau du sommet s. Plus précisément, la banque génère $(n+1)$ paires $(\mathsf{sk}_0^{(i)}, \mathsf{pk}_0^{(i)})$ et publie les signatures $\tau_s \leftarrow \Sigma_0.\texttt{Sign}(\mathsf{sk}_0^{(|s|)}, (g_s, h_s))$, pour tout $s \in \mathcal{S}_n$, dans sa clé publique. Pour prouver que la paire (g_s, h_s) est valide pour une dépense de 2^ℓ, l'utilisateur prouve l'existence d'une signature τ_s dessus sous la clé publique $\mathsf{pk}_0^{(n-|s|)}$. Si cette solution n'augmente que très légèrement la complexité d'une dépense, elle implique de devoir fournir, dans la clé publique bpk, 2^{n+1} signatures et $(n+1)$ clés publiques $\mathsf{pk}_0^{(i)}$.

Il est possible cependant de réduire la taille de ces données publiques en notant que, pour tout $f \in \mathcal{L}(|s|)$, il existe une feuille $\ell \in \mathcal{F}_n$ telle que $s||f = \ell$. Par conséquent, les relations $e(g_s, \widetilde{g}_{|s|,f}) = e(g_\ell, \widetilde{g})$ et $e(h_s, \widetilde{g}_{|s|,f}) = e(h_\ell, \widetilde{g})$ sont vérifiées, ce qui ouvre la voie à une autre manière de prouver la validité d'une paire (g_s, h_s). En effet, plutôt que de prouver que celle-ci est valide, on va prouver que (g_ℓ, h_ℓ) est valide et que ces relations sont vérifiées. La banque n'a alors besoin que d'une paire de clés $(\mathsf{sk}_0, \mathsf{pk}_0)$ avec laquelle elle ne signera que les paires associées aux feuilles de l'arbre. Cela entraine, certes, un surcout lors d'une dépense (car il y a davantage de relations à prouver) mais permet de diminuer sensiblement les paramètres que l'utilisateur doit conserver. Il s'agit donc d'un compromis qui peut se révéler intéressant lorsque le système est implémenté sur un périphérique disposant d'un espace de stockage limité.

Les résultats concernant la sécurité du système de monnaie électronique présenté dans cette section sont énoncés par le théorème suivant.

Théorème 34. *Dans le modèle standard, et en supposant que H est une fonction de hachage résistante aux collisions, notre système de monnaie électronique divisible est anonyme sous les hypothèses* SXDH *et* $\mathsf{EMDDH}-2$*, traçable si Σ_0 est un schéma de signature EUF-SCMA sûr et si Σ_1 est un schéma de signature EUF-CMA sûr, et satisfait la propriété de non-diffamation sous l'hypothèse* $q-\mathsf{SDH}$ *si Σ_{ots} est un schéma de signature à usage unique dont la sécurité est forte.*

Remarque 12. De même que pour la construction précédente, il est possible de reposer sur une hypothèse plus faible, à savoir $\mathsf{EMDDH}-1$, si un niveau inférieur d'anonymat est suffisant.

4.4.3 Preuves de Sécurité

Nous prouvons ici le théorème 34. Les preuves de la traçabilité et de la propriété de non-diffamation sont similaires à celles proposées dans la section 4.3. Nous les adaptons cependant à cette nouvelle construction afin d'être complet.

Preuve de la traçabilité

L'objectif de l'adversaire $\mathcal{A}$ est de produire, après q_w retraits, u traces valides de transactions $\{(2^{\ell_i}, Z_i, \Pi_i)\}_{i=1}^u$ telles que $\texttt{Identify}((2^{\ell_i}, Z_i, \Pi_i), (2^{\ell_j}, Z_j, \Pi_j)) = \perp$ pour tout $i \neq j$ et $\sum_{i=1}^u 2^{\ell_i} > q_w \cdot 2^w$. Supposons que $\mathcal{A}$ réussisse, nous distinguons les 3 cas suivants.

- Cas 1 : $\exists i$ tel que Π_i contienne une mise en gage d'une paire (g_s, h_s) qui n'a jamais été signée par la banque lors de l'exécution de l'algorithme $\texttt{BKeygen}$.
- Cas 2 : $\exists i$ tel que Π_i contienne une mise en gage d'une paire $(u_1^{\mathsf{usk}}, u_2^x)$ qui n'a jamais été signée par la banque lors d'une requête $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$.
- Cas 3 : $\forall 1 \leq i \leq u$, il existe une signature $\tau_s \in \mathsf{bpk}$ sur la paire (g_s, h_s) mise en gage dans Π_i et les paires $(u_1^{\mathsf{usk}}, u_2^x)$ impliquées dans ces traces ont toute été signées lors de requêtes $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$.

Considérons tout d'abord le troisième cas. Soient $x_1, \ldots, x_{q_w}$ les scalaires associés aux pièces retirées lors des requêtes $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$. Puisqu'un montant $\sum_{i=1}^u 2^{\ell_i} > q_w \cdot 2^w$ a été déposé, la banque a calculé $\sum_{i=1}^u 2^{\ell_i}$ éléments z_j. Ceux-ci appartenant tous (dans le cas 3) à l'ensemble $\{G^{x_i \cdot y_\epsilon} \prod_{j=1}^n y_{j,b_j}\}_{b_1 \ldots b_n \in \mathcal{F}_n, 1 \leq i \leq q_w}$ qui est de taille $q_w \cdot 2^n$, il existe au moins une paire (i, j), avec $i \neq j$ telle que $z_i = z_j$. Cependant, comme nous l'avons expliqué dans la description de l'algorithme $\texttt{Identify}$, une telle collision signifie, sauf avec une probabilité négligeable, qu'un même scalaire x a été utilisé lors de ces deux transactions. Le scalaire x étant lié à une unique clé publique upk, celle-ci aurait alors été retournée par l'algorithme $\texttt{Identify}$ (car H est résistante aux collisions). Par conséquent, il est très improbable que le cas 3 survienne.

Lemme 35. *Un adversaire $\mathcal{A}$ produisant des traces satisfaisant le cas 1 avec une probabilité ϵ peut être converti en un adversaire contre la sécurité EUF-SCMA du schéma Σ_0 réussissant avec une probabilité $\epsilon/(n+1)$.*

Preuve. La réduction $\mathcal{R}$ génère normalement les paramètres publics et choisit un indice $i^* \in [0, n]$. Elle envoie alors les paires (g_s, h_s), pour tout $s \in \mathcal{S}_n$ tel que $|s| = i^*$, au challenger du jeu EUF-SCMA qui lui retourne les signatures τ_s correspondantes ainsi qu'une clé publique pk. $\mathcal{R}$ peut alors générer les autres paires $(sk_0^{(i)}, \mathsf{pk}_0^{(i)})$ pour $i \neq i^*$ et les utilise pour signer les autres sommets. Finalement, il définit $\mathsf{pk}_0^{(i^*)} = \mathsf{pk}$ et publie $\{\mathsf{pk}_0^{(i)}\}$ ainsi que les signatures τ_s. Puisque la clé $\mathsf{sk}_0^{(i)}$ n'est plus impliquée dans le reste du protocole, $\mathcal{R}$ est capable de gérer toutes les requêtes émises par $\mathcal{A}$. À la fin, celui-ci retourne u traces dont l'une d'entre elles contient une mise en gage vers une paire (g_s, h_s) que la banque n'a pas signée. La validité du système de preuve utilisé assure alors que les traces contiennent également une mise en gage d'un élément τ_s tel que $\Sigma_0.\mathtt{Verify}((g_s, h_s), \tau_s, \mathsf{pk}_0^{(n-\ell)}) = 1$. Si $\ell \neq i^*$, alors $\mathcal{R}$ arrête. Sinon, τ_s constitue une contrefaçon valide sur (g_s, h_s) pour la clé pk et peut donc être utilisé pour casser la sécurité du schéma Σ_0.

Lemme 36. *Un adversaire $\mathcal{A}$ produisant des traces satisfaisant le cas 2 avec une probabilité ϵ peut être converti en un adversaire contre la sécurité EUF-CMA du schéma Σ_1 réussissant avec la même probabilité.*

Preuve. La réduction $\mathcal{R}$ génère normalement les paramètres publics et la clé de la banque à l'exception de pk_1 qui est définie comme étant pk, la clé reçue du challenger $\mathcal{C}$ du jeu définissant la sécurité EUF-CMA. $\mathcal{R}$ est alors capable de répondre directement à toutes les requêtes de sécurité à l'exception de celles de type $\mathcal{O}\mathtt{Withdraw}_{\mathcal{B}}$ pour lesquelles il transmet les paires $(u_1^{\mathsf{usk}}, u_2^x)$ à $\mathcal{C}$ afin d'obtenir les signatures σ correspondantes.

A la fin du jeu, $\mathcal{A}$ retourne u traces de transactions dont l'une d'entre elles contient une mise en gage d'une paire $(u_1^{\mathsf{usk}}, u_2^x)$ jamais signée par la banque. Là encore, la validité du système de preuve implique la mise en gage de σ tel que $\Sigma_1.\mathtt{Verify}((u_1^{\mathsf{usk}}, u_2^x), \sigma, \mathsf{pk}_1) = 1$. Cet élément peut donc être extrait et utilisé pour casser la sécurité EUF-CMA de Σ_1.

Preuve de la propriété de non-diffamation

Un adversaire $\mathcal{A}$ remportant le jeu de la non-diffamation est capable de produire deux traces accusant un utilisateur honnête upk de double-dépense. Cela signifie (sinon upk ne serait pas honnête) qu'au moins l'une d'entre elles n'a pas été produite par $\mathcal{R}$. Soit pk'_{ots} la clé publique du schéma de signature à usage unique utilisée dans la trace falsifiée, nous distinguons les deux cas suivants :

- Cas 1 : pk'_{ots} est l'une des clés que $\mathcal{R}$ a utilisées pour répondre aux requêtes $\mathcal{O}\mathtt{Spend}$.
- Cas 2 : pk'_{ots} n'a jamais été utilisée par $\mathcal{R}$.

Si $\mathcal{A}$ produit des traces rentrant dans le cas 1, alors il peut être converti en un adversaire contre la sécurité de Σ_{ots}. En effet, ce schéma de signature est utilisé pour signer l'intégralité de la trace. Par conséquent, une trace falsifiée nécessitera une nouvelle signature η qui sera donc une contrefaçon pour Σ_{ots}.

Lemme 37. *Soit q_s (resp. q_a) une borne sur le nombre de requêtes $\mathcal{O}\mathtt{Spend}$ (resp. $\mathcal{O}\mathtt{Add}$). Un adversaire produisant des traces satisfaisant le cas 2 avec probabilité ϵ peut être transformé en un adversaire contre l'hypothèse $q_s - \mathsf{SDH}$ réussissant avec probabilité ϵ/q_a.*

Preuve. Soit $(g, g^{\alpha}, \ldots, g^{\alpha_{q_s}})$ un challenge $q_s - \mathsf{SDH}$. $\mathcal{R}$ choisit un indice $i^* \in [1, q_a]$ et génère normalement les paramètres publics sauf u_1 qu'il définit comme étant g^z pour un certain scalaire z. De plus, elle calcule q_s paires de clés $(\mathsf{sk}_{ots}^{(i)}, \mathsf{pk}_{ots}^{(i)}) \leftarrow \Sigma_{ots}.\mathtt{Keygen}(1^k)$ et définit $w \leftarrow g^{\prod_{i=1}^{q_s}(\alpha + H(\mathsf{pk}_{ots}^{(i)}))}$ (ce qui peut être calculé à partir du challenge $q_s - \mathsf{SDH}$). $\mathcal{R}$ peut alors répondre aux différentes requêtes de la manière suivante.

- $\mathcal{O}\texttt{Add}()$: lorsque $\mathcal{A}$ émet la i-ème requête $\mathcal{O}\texttt{Add}$, $\mathcal{R}$ exécute l'algorithme $\texttt{Keygen}$ si $i \neq i^*$ et définit $\mathsf{upk}^* \leftarrow g^{\alpha}$ dans le cas contraire.
- $\mathcal{O}\texttt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ retourne la clé secrète si $\mathsf{upk} \neq \mathsf{upk}^*$. Sinon, elle arrête.
- $\mathcal{O}\texttt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ conserve la clé publique upk qui est maintenant considérée comme enregistrée.
- $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}(\mathsf{bsk}, \mathsf{upk})$: $\mathcal{R}$ agit normalement si $\mathsf{upk} \neq \mathsf{upk}^*$ et simule la preuve interactive de connaissance de α dans le cas contraire.
- $\mathcal{O}\texttt{Spend}(\mathsf{upk}, 2^{\ell})$: $\mathcal{R}$ agit normalement si $\mathsf{upk} \neq \mathsf{upk}^*$. Sinon, pour répondre à la $j-$ème requête sur upk^*, elle commence par sélectionner la paire $(sk_{ots}^{(j)}, pk_{ots}^{(j)})$ et calcule l'élément $\mu \leftarrow g^{\prod_{i=1, i\neq j}^{q_s}(\alpha + H(pk_{ots}^{(i)}))}$ qui vérifie $\mu = w^{\frac{1}{\alpha + H(pk_{ots}^{(j)})}}$. Elle peut alors utiliser $sk_{ots}^{(j)}$ comme spécifié dans le protocole $\texttt{Spend}$.

$\mathcal{A}$ finit par retourner deux traces valides $(2^{\ell_1}, Z_1, \Pi_1)$ et $(2^{\ell_2}, Z_2, \Pi_2)$ accusant upk de double-dépense. Si $\mathsf{upk} \neq \mathsf{upk}^*$, alors $\mathcal{R}$ arrête. La validité du système de preuve utilisé implique que la trace falsifiée a été signée en utilisant une clé sk_{ots} et donc qu'elle contient une mise en gage de $\mu = w^{\frac{1}{\alpha + H(\mathsf{pk}_{ots})}}$. Puisque nous sommes dans le cas 2, $pk_{ots} \notin \{pk_{ots}^{(i)}\}_i$, et donc $H(pk_{ots}) \notin \{H(pk_{ots}^{(i)})\}_i$ car la fonction H est résistante aux collisions. L'élément μ peut alors être utilisé pour résoudre le problème q_s-SDH dans $\mathbb{G}_1$ (comme expliqué dans [BB08]). $\mathcal{R}$ est donc capable de résoudre ce problème puisque le jeu ne sera pas arrêté avec probabilité $1/q_a$.

Preuve de l'anonymat

L'objectif d'un adversaire $\mathcal{A}$ contre l'anonymat est d'être capable de distinguer, parmi deux utilisateurs de son choix, lequel est impliqué dans une dépense. Nous construisons ici une réduction $\mathcal{R}$ prouvant qu'un tel adversaire peut être utilisé contre l'hypothèse $\mathsf{EMDDH} - 2$.

Soient $(g, g^a, g^t, \{g^{y^i}\}_{i=1}^{i=n}, \{g^{t\cdot y^i}\}_{i=1}^{i=n}, \{g^{x\cdot y^i}\}_{i=0}^{i=n-1}, \{g^{x\cdot t\cdot y^i}\}_{i=0}^{i=n-1}, \{\widetilde{g}^{y^{-i}}\}_{i=0}^{i=n}, \widetilde{g}^a) \in \mathbb{G}_1^{4n+3} \times \mathbb{G}_2^{n+2}$ et $(g^{z_1}, g^{z_2}) \in \mathbb{G}_1^2$ un challenge $\mathsf{EMDDH} - 2$. La réduction $\mathcal{R}$ choisit un niveau d^* aléatoire ainsi qu'un sommet $s^* = b_1^* ... b_{|s^*|}^*$ de ce niveau (*i.e.* $|s^*| = d^*$). Dans toute la simulation, $\mathcal{R}$ va agir comme si $y_\epsilon = y^n$, $a_{d^*} = a$ et, pour $1 \leq i \leq d^*$, $y_{i,b_i^*} = u_{i,b_i^*} \xleftarrow{\$} \mathbb{Z}_p$ et $y_{i,\bar{b}_i^*} = y^{-1} \cdot u_{i,\bar{b}_i^*}$ avec $u_{i,\bar{b}_i^*} \xleftarrow{\$} \mathbb{Z}_p$. Pour tout $s = b_\ell \dots b_{|s|}$, nous définissons l'entier $d_s = |\{\ell \leq i \leq min(|s|, d^*) : b_i \neq b_i^*\}|$. Pour générer les paramètres publics, $\mathcal{R}$

- définit $(h, u_1, u_2) \leftarrow (g^t, g^{d_1}, g^{d_2})$ pour des scalaires aléatoires $d_1, d_2 \xleftarrow{\$} \mathbb{Z}_p$;
- définit $g_\epsilon \leftarrow g^{y^n}$;
- choisit, pour $1 \leq i \neq d^* \leq n$ $a_i \xleftarrow{\$} \mathbb{Z}_p$ et calcule $g_i \leftarrow g^{a_i}$;
- définit $g_{d^*} \leftarrow g^a$;
- choisit, pour $1 \leq i \leq n$, $u_{i,0}, u_{i,1} \xleftarrow{\$} \mathbb{Z}_p$ et calcule

$$(g_s, h_s) \leftarrow ((g^{y^{n-d_s}})^{\prod_{i=1}^{|s|} u_{i,b_i}}, (g^{t\cdot y^{n-d_s}})^{\prod_{i=1}^{|s|} u_{i,b_i}}), \quad \forall s = b_1 \dots b_{|s|};$$

- calcule, pour tout $0 \leq i \neq d^* \leq n$ et tout $f \in \mathcal{L}_i$,

$$(\widetilde{g}_{i,f}, \widetilde{h}_{i,f}) \leftarrow ((\widetilde{g}^{y^{-d_f}})^{\prod_{j=i+1}^{n} u_{j,b_j}}, \widetilde{g}_{i,f}^{-a_i});$$

- calcule, pour tout $f \in \mathcal{L}_{|s^*|}$,

$$(\widetilde{g}_{d^*,f}, \widetilde{h}_{d^*,f}) \leftarrow (\widetilde{g}^{\prod_{j=d^*+1}^{n} u_{j,b_j}}, (\widetilde{g}^a)^{-\prod_{j=d^*+1}^{n} u_{j,b_j}}).$$

Finalement, $\mathcal{R}$ génère une chaîne de référence commune CRS de manière à obtenir des preuves satisfaisant l'indistinguablibité des témoins. Une telle CRS ne peut être distinguée de celle générée normalement par l'algorithme $\texttt{Setup}$, sous l'hypothèse SXDH.

Le point essentiel de cette réduction est que, pour tout s qui n'est pas un préfixe ou un suffixe de s^*, nous avons $d_s > 0$. Par conséquent, l'élément g_s (resp. h_s) qui lui est associé est égal à $g^{y^j \cdot r_s}$ (resp. $g^{t \cdot y^j \cdot r_s}$) pour un certain $j < n$ et un certain $r_s \in \mathbb{Z}_p$ connu de $\mathcal{R}$. Celle-ci peut alors utiliser s pour répondre aux requêtes $\mathcal{O}\mathtt{Spend}$ puisque le challenge $\mathsf{EMDDH}-2$ contient les paires $(g^{x \cdot y^i}, g^{x \cdot t \cdot y^i})$ pour $i < n$.

Soit q_w une borne sur le nombre de requêtes $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$, $\mathcal{R}$ choisit un $i^* \in [0, q_w]$ et répond aux différentes requêtes de la manière suivante.

- $\mathcal{O}\mathtt{Add}()$: $\mathcal{R}$ exécute l'algorithme $\mathtt{Keygen}$ et retourne upk (ou mpk).
- $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}(\mathsf{bsk}, \mathsf{upk})$: lors de la i-ème requête $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$, $\mathcal{R}$ agit normalement si $i \neq i^*$ et comme si le secret de la pièce retirée était x sinon (en envoyant $(g^x)^{d_2}$ et en simulant la preuve de connaissance, x n'étant pas connu). La clé publique impliquée dans ce dernier cas est alors notée upk^*.
- $\mathcal{O}\mathtt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ agit normalement si la requête ne porte pas sur upk^*. Sinon, elle arrête la simulation.
- $\mathcal{O}\mathtt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ conserve cette clé publique qui est maintenant considérée comme enregistrée.
- $\mathcal{O}\mathtt{Spend}(\mathsf{upk}, 2^\ell)$: $\mathcal{R}$ peut répondre à toutes les requêtes si $\mathsf{upk} \neq \mathsf{upk}^*$ car elle connait alors tous les secrets associés. Sinon, la réduction reste capable de répondre tant que $c_{\mathsf{upk}^*} \leq m_{\mathsf{upk}^*} \cdot 2^n - 2^{n-d^*} - 2^\ell$ (autrement, elle doit arrêter) puisque cette condition implique qu'il existe au moins un sommet non dépensé s (de profondeur $n - \ell$) qui n'est ni un préfixe, ni un suffixe de s^*. Par conséquent, $d_s > 0$ ce qui rend possible, comme nous l'avons expliqué précédemment, le calcul de g_s^x et h_s^x. $\mathcal{R}$ peut alors retourner des paires valides $t_s \leftarrow (g^{r_1}, g_s^x \cdot g_{n-\ell}^{r_1})$ et $v_s \leftarrow (g^{r_2}, (\mathsf{upk}^*)^R \cdot h_s^x \cdot g_{n-\ell}^{r_2})$ où $R \leftarrow H(info)$ et simuler la preuve non-interactive (ce qui est possible avec le type de CRS généré).

Durant la phase de challenge, $\mathcal{A}$ retourne $\{\mathsf{upk}_0, \mathsf{upk}_1\}$ ainsi qu'un montant 2^ℓ. Il est bien sûr exigé que ces utilisateurs n'aient pas déjà dépensé un montant supérieur à $m_{\mathsf{upk}_b} \cdot 2^n - 2^\ell$, c'est-à-dire qu'il leur reste suffisamment de réserve pour cette nouvelle dépense. Si $\mathsf{upk}^* \notin \{\mathsf{upk}_0, \mathsf{upk}_1\}$ ou $\ell \neq n - d^*$ (*i.e.* s^* ne correspond pas à la valeur demandée 2^ℓ) alors $\mathcal{R}$ arrête la simulation. Sinon, $\mathcal{R}$ choisit deux scalaires aléatoires $r, r' \xleftarrow{\$} \mathbb{Z}_p$ et retourne, en même temps que des preuves simulées,

$$((g^{z_1})^{-\prod_{i=1}^{|s^*|} u_{i,b_i^*}} \cdot g^r, g^{a \cdot r}) \text{ and } ((g^{z_2})^{-\prod_{i=1}^{|s^*|} u_{i,b_i^*}} \cdot g^{r'}, (\mathsf{upk}^*)^R \cdot g^{a \cdot r'})$$

- dans le cas où $(z_1, z_2) = (x \cdot y^n / a, x \cdot t \cdot y^n / a)$ alors
  - la première paire est $(g^{r_1}, g_{s^*}^x \cdot g_{|s^*|}^{r_1})$ avec $r_1 = r - z_1 \cdot \prod_{i=1}^{|s*|} u_{i,b_i^*}$:

$$\begin{aligned} g_{s^*}^x \cdot g_{|s^*|}^{r_1} &= g^{(x \cdot y^n) \prod_{i=1}^{|s*|} u_{i,b_i^*}} \cdot g^{a \cdot (r - z_1 \cdot \prod_{i=1}^{|s*|} u_{i,b_i^*})} \\ &= g^{(x \cdot y^n) \prod_{i=1}^{|s*|} u_{i,b_i^*}} \cdot g^{a \cdot r} \cdot g^{(-x \cdot y^n) \prod_{i=1}^{|s*|} u_{i,b_i^*}} = g^{a \cdot r} \end{aligned}$$

  - la seconde est $(g^{r_2}, (\mathsf{upk}^*)^R \cdot h_{s^*}^x \cdot g_{|s^*|}^{r_2})$ avec $r_2 = r' - z_2 \cdot t \prod_{i=1}^{|s*|} u_{i,b_i^*}$:

$$\begin{aligned} h_{s^*}^x \cdot g_{|s^*|}^{r_2} &= h_{s^*}^x \cdot g_{|s^*|}^{-z_2 \cdot t \prod_{i=1}^{|s*|} u_{i,b_i^*}} \cdot g_{|s^*|}^{r'} \\ &= (g_{s^*}^x \cdot g_{|s^*|}^{-z_2 \cdot \prod_{i=1}^{|s*|} u_{i,b_i^*}})^t \cdot g^{a \cdot r'} = g^{a \cdot r'} \end{aligned}$$

    Par conséquent, chacune des paires retournées par $\mathcal{R}$ est valide.
- dans le cas où (z_1, z_2) est aléatoire, ces paires sont respectivement égales à $(g^\alpha \cdot g^r, g^{a \cdot r})$ et $(g^\beta \cdot g^{r'}, (\mathsf{upk}^*)^R \cdot g^{a \cdot r'})$, et masque parfaitement upk^*.

Le bit renvoyé par $\mathcal{A}$ peut alors être utilisé pour résoudre le problème $\mathsf{EMDDH}-2$.

La simulation se poursuivra jusqu'à la fin, à condition que $\mathcal{R}$ ait correctement deviné la valeur dépensée lors de la phase de challenge (*i.e.* $\ell = n - d^*$) et que la pièce retirée durant la i^*-ème

requête appartienne à upk_0 ou upk_1. La probabilité que $\mathcal{R}$ ne s'arrête pas est donc au moins supérieure à $2/((n+1) \cdot q_w)$.

Remarque 13. Là encore, les éléments du challenge $\mathsf{EMDDH}-2$ dont l'exposant est un multiple de t ne servent qu'à simuler les éléments v_s lors des dépenses. Ceux-ci n'intervenant pas dans une construction non-corrélable (même fortement), il est alors possible de faire reposer la sécurité de cette dernière sur l'hypothèse $\mathsf{EMDDH}-1$, qui est plus faible.

4.5 Comparaison des Performances

Nous comparons, dans le tableau 4.1, les performances de nos constructions avec celles du schéma de Canard et Gouget [CG10] qui est le plus efficace de l'état de l'art. Néanmoins, il est nécessaire de détailler davantage certains points de cette comparaison, notamment en raison de l'utilisation de groupes de différents types.

La construction de [CG10] nécessite d'utiliser des sous-groupes de $\mathbb{Z}_{r_i}$, pour certains premiers r_i de taille approximative $|q|$, ainsi que des groupes bilinéaires de même ordre. Pour un niveau de sécurité de 128 bits, l'article [GPS08] donne les valeurs suivantes : $|p| = |\mathbb{G}_1| = 256$, $|\mathbb{G}_2| = 512$ et $|q| = |\mathbb{G}_T| = 3072$ si les courbes Barreto-Naehrig [BN05] sont utilisées.

Paramètres publics. A première vue, les paramètres publics de nos constructions peuvent sembler très importants. Cependant, pour $n = 10$ (ce qui permet de diviser la pièce en 1024 parts), l'espace de stockage qu'ils nécessitent est de 3.1 Mo pour [CG10], 1.1 Mo pour notre première construction (section 4.3) et 525 Ko pour la seconde (section 4.4).

Dépenses et Retraits. Nos constructions sont les premières à proposer des protocoles `Spend` et `Withdraw` exécutables en temps constants. Cela est d'autant plus important qu'il s'agit des protocoles dans lesquels interviennent les utilisateurs et donc pour lesquels il est indispensable d'offrir de bonnes performances. En effet, un système imposant à ses utilisateurs d'attendre plusieurs minutes pour effectuer une dépense ou un retrait aurait peu de chance d'être adopté. Par ailleurs, on peut remarquer que l'espace requis pour enregistrer les pièces est très limité pour nos deux constructions. D'un point de vue efficacité, notre premier système offre de meilleures performances lors d'une dépense mais souffre d'une complexité nettement supérieure lors d'un dépôt. Il est possible de supprimer les 2 couplages (« `Pair` ») dans le protocole `Spend` de notre deuxième construction mais au prix d'une augmentation de la taille des paramètres publics, comme l'explique la remarque 11 de la section 4.4.2.

Dépôt. Le principal problème de notre première construction est le coût de calcul et de stockage d'un dépôt. En effet, il est nécessaire, quel que soit le montant déposé, d'effectuer 2^n couplages et de conserver leurs résultats. Bien que certaines astuces, telles que le fait de ne stocker qu'une empreinte des éléments calculés, permettent de réduire la taille des données conservées, une utilisation à grande échelle de ce système semble difficile. Notre deuxième construction résout ce problème en éliminant tous les calculs inutiles et en ne conservant que le nombre minimal de numéros de série.

Schémas	[CG10]	Nous (section 4.3)	Nous (section 4.4)
Modèle	ROM	ROM/standard	ROM/standard
Paramètres Publics	$2^{n+3}\|q\| + 1\ pk$	$(n+2)\ pk + (2^{n+2}+3)\mathbb{G}_1$ $+ (1+(n+1)2^n)\ \mathbb{G}_2 +$ $(2^{n+1}-1)\ \|\texttt{Sign}\|$	$2\ pk + 2^n\ \|\texttt{Sign}\|$ $+ (2^{n+2}-1)\ \mathbb{G}_2$ $+ (2^{n+2}+n+4)\ \mathbb{G}_1$
Complexité `Withdraw`	$(2^{n+3}+2^{n+2}-5)$`exp` $+ (n+2)$ `Sign`	1 `Sign`	1 `Sign`
Taille d'une pièce	$(2^{n+2}+n+1)\ \|q\|$ $+ (n+2)\ \|\texttt{Sign}\|$	$2\ \|p\| + \|\texttt{Sign}\|$	$2\ \|p\| + \|\texttt{Sign}\|$
Complexité `Spend`	$NIZK\{$ 3 `exp`* + 2 `Sign` + 2 `Pair` $\}$ + 1 `exp`	$NIZK\{$ 2 `exp` + 2 `Sign` $\}$ + 3 `exp` + 1 `Sign`	$NIZK\{$ 4 `exp` + 2 `Sign` + 2 `Pair` $\}$ + 7 `exp` + 1 `Sign`
Taille d'une trace	$3\ \|q\| + \|NIZK\|$	$2\ \mathbb{G}_1 + 1\ \|\texttt{Sign}\|$ $+\|NIZK\|$	$4\ \mathbb{G}_1 + 1\ \|\texttt{Sign}\| + 1\ \mathbb{G}_2$ $+\ \|NIZK\|$
Complexité `Deposit`	2^{l+1}`exp`	2^n `Pair`	2^{l+1} `Pair`
Taille `Deposit`	$2^l\ \|q\| + \|\texttt{Spend}\|$	$2^n\ \mathbb{G}_T + \|\texttt{Spend}\|$	$2^l\ \mathbb{G}_T + \|\texttt{Spend}\|$

Table 4.1 – Comparaison des performances de nos constructions avec celles de l'état de l'art pour des pièces d'une valeur 2^n et des dépenses (`Spend`) et dépôts (`Deposit`) d'un montant 2^l ($l \leq n$). La taille et la complexité d'une dépense correspondent au point de vue de l'utilisateur. `exp` fait référence au coût d'une exponentiation, `Pair` à celui d'un couplage, `Sign` à celui de l'émission d'une signature dont la clé publique est pk. $NIZK\{$`exp`$\}$ représente le coût d'une preuve non interactive d'une équation à exponentiations multiples, $NIZK\{$`Pair`$\}$ à celui d'une équation de produits de couplages et $NIZK\{$`Sign`$\}$ au coût d'une preuve de connaissance d'une signature. Pour finir, $NIZK\{$`exp`$^*\}$ correspond au coût d'une preuve d'égalité de logarithmes discrets dans des groupes d'ordre différent.

Chapitre 5

Techniques Cryptographiques pour l'Authentification Anonyme

Sommaire

Nous présentons dans ce chapitre certaines techniques cryptographiques pour l'authentification anonyme. Nous commençons par une primitive majeure, la signature de groupe, qui est à la base de plusieurs autres constructions telles que les attestations anonymes. Cependant, celles-ci n'étant pas adaptées à tous les cas d'usages, nous décrivons une nouvelle primitive offrant plus de flexibilité, notamment dans les procédures de levée d'anonymat. Cette dernière a été introduite dans l'article *Direct Anonymous Attestation with Dependent Basename Opening* [DLST14] co-signés avec Nicolas Desmoulins, Roch Lescuyer et Jacques Traoré et publié à la conférence CANS 2014.

5.1 Authentification Anonyme

5.1.1 Signature de Groupe

Le paiement anonyme, que nous avons étudié dans le chapitre précédent, n'est en fait qu'un cas particulier des problèmes d'authentification anonyme qui se posent dans une société numérique. En effet, lors d'un paiement, l'acheteur doit essentiellement s'authentifier comme l'un des utilisateurs du système qui a retiré une pièce. Celui-ci est alors anonyme au sein de cet ensemble. Le reste, tel que le calcul des numéros de série, n'intervient qu'a posteriori et vise à prévenir les abus propres à ce cas particulier.

Face aux différents cas d'usage relatifs à l'authentification anonyme, la cryptographie moderne a fait preuve d'une remarquable flexibilité en proposant de nouvelles primitives adaptées à chacun d'eux. L'une des plus fameuses d'entre elles est probablement la *signature de groupe*, introduite par Chaum et Van Heyst [Cv91]. Cette primitive considère un groupe d'utilisateurs géré par une entité, appelée *gestionnaire du groupe*, contrôlant l'adhésion de nouveaux membres. Chacun de ces utilisateurs dispose de la capacité de signer au nom du groupe, c'est-à-dire d'émettre une signature

dont la validité pourra être publiquement testée mais dont l'auteur ne pourra être identifié. En émettant une signature, l'utilisateur s'authentifie donc comme un membre du groupe mais ne révèle aucune autre information sur son identité. Il est alors anonyme au sein du groupe.

Ainsi définie, cette primitive semble plutôt simple à réaliser. Il suffit par exemple que tous les membres du groupe utilisent la même clé secrète pour un schéma de signature numérique. La construction obtenue est évidemment anonyme mais soulève quelques problèmes de sécurité. En effet, une fois la clé secrète obtenue, l'utilisateur ne peut plus être identifié, ce qui peut conduire à certaines dérives (revente de la clé secrète, utilisation abusive de celle-ci, etc...). Pour éviter cela, les schémas de signature de groupe définissent également une entité, appelée *autorité d'ouverture*, disposant du pouvoir de lever l'anonymat. Celle-ci exclut par conséquent le type de construction que nous venons de décrire car chaque utilisateur doit alors disposer d'une clé secrète spécifique permettant à cette autorité de distinguer les signatures qu'il a émises.

Considérons par exemple le système de transport public d'une grande ville. On trouve d'un côté l'opérateur du service qui souhaite s'assurer que seuls les utilisateurs légitimes accèdent au réseau de transport et de l'autre les usagers qui peuvent souhaiter que leurs mouvements ne soient pas tracés. Ces points de vue, souvent présentés comme incompatibles, peuvent en fait être conciliés grâce à la signature de groupe. En effet, supposons que l'opérateur agisse comme le gestionnaire du groupe. Lorsqu'un utilisateur souscrit à un abonnement, il obtient une clé secrète lui permettant d'émettre des signatures de groupe. A chaque fois qu'il accède au service, il signe un message aléatoire (pour éviter la réutilisation de la signature par d'autres usagers) reçu de la borne, s'authentifiant ainsi comme un abonné en règle. Les propriétés des signatures de groupe garantissent alors le respect de la vie privée de l'utilisateur qui est anonyme parmi les abonnés. Afin de respecter d'éventuelles contraintes légales, le pouvoir de lever l'anonymat (c'est-à-dire celui de l'autorité d'ouverture) peut être confié à une autorité judiciaire qui pourra, par exemple, l'utiliser pour identifier les témoins d'un certain événement.

Suite à l'introduction de cette nouvelle primitive, de nombreuses constructions furent proposées (*e.g.* [CP95, Cam97, CS97]) mais souffrant toutes de certaines vulnérabilités, notamment face aux coalitions d'utilisateurs malveillants. En 2000, Ateniese *et al* [ACJT00] publièrent le premier schéma efficace résistant à l'ensemble des menaces identifiées. Cependant, comme le firent remarquer Bellare, Micciancio et Warinschi [BMW03], ce résultat était à relativiser en raison de l'absence de définitions formelles des propriétés de sécurité à satisfaire ainsi que les attaques contre lesquelles se prémunir. Ils proposèrent alors le premier modèle de sécurité pour les signatures de groupes *statiques*, dont l'ensemble des membres est fixé lors de la génération des paramètres. Celui-ci fut ensuite étendu aux groupes *dynamiques*, acceptant de nouveaux adhérents après la génération des paramètres, par des travaux indépendants [BSZ05, KY06].

Le modèle défini dans [BSZ05] considère trois propriétés de sécurité essentielles pour cette primitive. La première est la *traçabilité* qui stipule que seul un membre du groupe peut émettre une signature et que celui-ci doit pouvoir être identifié par l'autorité d'ouverture. La deuxième est la *non-diffamation* qui précise qu'un utilisateur ne peut être accusé à tort d'avoir émis une signature. La dernière est l'*anonymat* qui exige que seule l'autorité d'ouverture puisse identifier l'auteur d'une signature.

Cependant la définition de l'anonymat retenue dans [BSZ05] est particulièrement forte, car même la connaissance d'une clé secrète ne doit permettre d'identifier les signatures émises à l'aide de celle-ci. La construction d'un protocole de signature de groupe nécessite alors l'utilisation d'un schéma de chiffrement [AW04], ce qui impacte négativement l'efficacité. Boneh et Shacham [BS04] considérèrent une version plus modérée de l'anonymat, appelée *anonymat désintéressé*, assurant les mêmes garanties aux utilisateurs, sauf dans le cas où leur clé secrète serait révélée. Cette propriété, qui n'implique plus le chiffrement IND-CPA comme le prouvèrent Bichsel *et al* [BCN+10], permet des constructions très efficaces offrant même la possibilité de révoquer des utilisateurs.

5.1.2 Attestation Anonyme

Malheureusement, l'anonymat d'une signature de groupe, que ce soit celui défini dans [BMW03, BSZ05] ou sa version *désintéressée* [BS04], peut se révéler inadapté dans certaines situations. En effet, reprenons l'exemple de notre réseau de transport public. Supposons qu'un utilisateur malhonnête récupère la clé secrète contenue dans sa carte d'abonnement et s'en serve pour fabriquer des cartes clonées. Leur utilisation sera alors indétectable en raison des propriétés des signatures de groupe. Cela illustre l'une des limites de cette primitive, et notamment de son autorité d'ouverture. Celle-ci dispose en effet du pouvoir d'identifier les auteurs d'abus mais pas de les détecter.

Il peut donc être nécessaire de disposer d'une version plus flexible de l'anonymat, telle que celle proposée par les *attestations anonymes* [BCC04]. Cette primitive, très proche des signatures de groupe, rajoute un élément aux signatures, appelé *marqueur*, qui permet de relier celles produites avec la même clé et le même marqueur. Dans le cas du transport public, on peut par exemple imposer aux utilisateurs d'utiliser un marqueur dépendant de l'heure. Ainsi, tous les usagers dans un certain intervalle de temps (par exemple, une heure) émettront des signatures en utilisant le même marqueur. Les clones utilisant tous la même clé, l'opérateur du service sera capable de détecter leur présence sans trop empiéter sur la vie privée des autres utilisateurs, qui ne seront tracés que sur cet intervalle de temps.

Comme les signatures de groupe, les attestations anonymes ont longtemps souffert de l'absence de définitions précises et de modèle de sécurité. L'un des exemples est la *Liste Noire* que ces systèmes considèrent, censée contenir l'ensemble des clés secrètes révoquées. En effet, il n'est pas précisé qui maintient cette liste et surtout comment rajouter une clé secrète dessus, sachant que seul l'utilisateur a connaissance de celle-ci. Comme le firent remarquer Bernhard *et al.* [BFG+13], cela signifie en pratique que seuls les utilisateurs sont capables de se révoquer. Ces auteurs proposèrent alors le premier modèle de sécurité pour cette primitive présentant de nombreuses similarités avec celui des signatures de groupe décrit dans [BSZ05].

Malheureusement, les attestations anonymes ne permettent pas de levée d'anonymat car elles ne considèrent pas d'autorité d'ouverture. Cela pose problème pour notre cas d'usage car détecter les clones ne suffit pas, il faut également pouvoir les identifier. Il est donc nécessaire de définir une version modifiée des attestations anonymes proposant cette fonctionnalité.

Cependant, en pratique, le pouvoir de lever l'anonymat doit être partagé. Sinon, cela revient simplement à transférer la capacité de tracer les utilisateurs de l'opérateur du service (comme c'est le cas actuellement) à une autre entité, l'autorité d'ouverture. Il est possible d'utiliser des techniques classiques de la cryptographie à seuil pour s'assurer qu'une entité seule ne puisse avoir ce pouvoir. Néanmoins, cela pose un réel problème concret. En effet, comment ouvrir efficacement un grand nombre de signatures (par exemple pour identifier tous les témoins potentiels d'un crime) avec ce type de protocoles ?

Nous proposons dans la prochaine section une nouvelle primitive, appelée *attestation anonyme avec ouverture dépendant du marqueur* (AA-ODM) (par analogie aux travaux menés sur les signatures de groupe dans [SEH+13, LJ14]), répondant à ce problème. Celle-ci est une variante des attestations anonymes où l'identification des utilisateurs est rendue possible et confiée à deux entités, l'autorité d'ouverture et l'autorité de contrôle. La première joue un rôle similaire à celui d'une autorité d'ouverture d'un schéma de signature de groupe, à ceci près qu'elle ne peut désormais identifier l'émetteur d'une signature associée à un marqueur *bsn* que si l'autorité de contrôle lui délivre un jeton correspondant à ce marqueur. Le point important est qu'aucune de ces deux entités n'est capable d'obtenir de l'information sur l'auteur d'une signature sans la collaboration de l'autre. L'intérêt de ce mode de partage est qu'il est désormais possible d'avoir recours à des entités de puissances différentes. En effet le coût de l'identification est très faible pour l'autorité de contrôle, et surtout ne dépend pas du nombre de signatures à ouvrir. Ce rôle peut donc être joué par n'importe quelle entité (la police par exemple), quelle que soit sa

puissance de calcul. Par exemple, dans le cas précédent où les témoins potentiels doivent être identifiés, l'autorité de contrôle peut autoriser la levée d'anonymat de tous les usagers sur un certain créneau horaire en émettant un unique jeton.

De même que les attestations anonymes, qui offraient plus de possibilités dans le contrôle de l'anonymat, cette nouvelle primitive permet donc plus de flexibilité dans la procédure d'ouverture. Elle est alors capable de s'adapter à des contraintes très fortes, surtout qu'elle peut être instanciée efficacement, comme le prouve la construction de la section 5.2.3.

5.2 Attestation Anonyme avec Ouverture Dépendant du Marqueur

Nous présentons dans cette section un modèle de sécurité pour cette nouvelle primitive ainsi qu'une instanciation que nous prouvons sûre dans le ROM. Ces résultats ont été publiés dans l'article *Direct Anonymous Attestation with Dependent Basename Opening* [DLST14] lors de la conférence CANS 2014.

5.2.1 Définition

Un schéma d'attestation anonyme dépendant du marqueur est défini par les algorithmes suivants, faisant intervenir quatre types d'entités : les utilisateurs, le manager du groupe, l'autorité d'ouverture et celle de contrôle.

$\texttt{Setup}(1^k)$: cet algorithme génère *p.p.*, la description des paramètres publics du système à partir d'un paramètre de sécurité k.

$\texttt{Keygen}(p.p.)$: cet algorithme retourne la description de deux registres **Reg** et **Sreg**, ainsi que les paires de clés $(\mathsf{tsk}, \mathsf{tpk})$ pour l'autorité de contrôle, $(\mathsf{osk}, \mathsf{opk})$ pour l'autorité d'ouverture et $(\mathsf{isk}, \mathsf{ipk})$ pour le gestionnaire du groupe. La clé publique du groupe est alors $\mathsf{gpk} \leftarrow (p.p., \mathsf{ipk}, \mathsf{tpk}, \mathsf{opk})$ qui sera implicitement prise en entrée par tous les autres algorithmes. On suppose par ailleurs que le registre **Reg** est publiquement accessible, tandis que le registre **Sreg** n'est connu que de l'autorité d'ouverture.

$\texttt{UKeygen}()$: cet algorithme génère une paire de clés $(\mathsf{sk}_i, \mathsf{pk}_i)$. La valeur pk_i est alors rendue publique, ce qui suppose que tout le monde peut en obtenir une copie authentique.

`Join` : ce protocole interactif s'exécute entre un nouvel utilisateur i dont l'entrée est sk_i et le gestionnaire du groupe, dont les entrées sont isk et **Reg**. Si le protocole aboutit, l'utilisateur obtient sa clé de signature de groupe gsk_i tandis que le gestionnaire de groupe met à jour le registre **Reg**.

$\texttt{Sign}(\mathsf{gsk}_i, m, bsn)$: cet algorithme prend en entrée la clé de signature gsk_i de l'utilisateur i, un message m et un marqueur bsn, et retourne une signature σ.

$\texttt{Verify}(\sigma, m, bsn)$: cet algorithme retourne 1 si σ est une signature valide sur m pour le marqueur bsn et 0 sinon.

$\texttt{Link}(\sigma, m, \sigma', m', bsn)$: cet algorithme retourne 1 si σ et σ' sont deux signatures valides pour un même marqueur bsn et ont été émises par le même utilisateur, et 0 sinon.

$\texttt{Token}(\mathsf{tsk}, bsn)$: cet algorithme prend en entrée la clé tsk de l'autorité de contrôle et un marqueur bsn et retourne le jeton τ correspondant.

$\texttt{Open}(\sigma, m, bsn, \tau, \mathsf{osk}, \mathbf{Reg}, \mathbf{Sreg})$: cet algorithme retourne l'indice i d'un utilisateur ainsi qu'une preuve π que celui-ci a bien produit la signature σ sur m pour le marqueur bsn, et $\perp$ autrement.

$\texttt{Judge}(i, \sigma, m, bsn, \tau, \pi, \mathbf{Reg})$: cet algorithme retourne 1 si π est une preuve valide que l'utilisateur i a émis σ et 0 sinon.

5.2.2 Modèle de Sécurité

Comme une signature de groupe, une attestation anonyme avec ouverture dépendant du marqueur doit vérifier les propriétés de *traçabilité* et de *non-diffamation*. La principale différence entre ces deux primitives concerne la définition de l'anonymat. En effet, la présence de deux entités disposant partiellement du pouvoir de lever celui-ci nécessite de définir deux propriétés distinctes, à savoir l'*anonymat vis-à-vis de l'autorité de contrôle* et celui *vis-à-vis de l'autorité d'ouverture*. Ces propriétés sont modélisées par des jeux de sécurité entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$ ayant accès à certains des oracles suivants.

- $\mathcal{O}\texttt{Add}()$ est un oracle permettant à $\mathcal{A}$ de rajouter un nouvel utilisateur i au système. $\mathcal{C}$ exécute alors l'algorithme $\texttt{UKeygen}$, conserve sk_i dans une liste $\mathcal{HK}$ et envoie la clé publique pk_i correspondante à $\mathcal{A}$. L'utilisateur i est alors considéré comme *honnête*.
- $\mathcal{O}\texttt{Corrupt}(i)$ est un oracle utilisé par $\mathcal{A}$ pour corrompre l'utilisateur honnête i. $\mathcal{C}$ envoie alors à $\mathcal{A}$ l'ensemble des données secrètes de i à savoir sk_i et, si elle existe, gsk_i. L'utilisateur i est désormais considéré comme *corrompu*.
- $\mathcal{O}\texttt{Join}_U(i)$ est un oracle qui exécute la partie utilisateur du protocole $\texttt{Join}$. Cet oracle est utilisé par $\mathcal{A}$ jouant le rôle du gestionnaire du groupe contre l'utilisateur honnête i.
- $\mathcal{O}\texttt{Join}_M(i)$ est un oracle qui exécute la partie gestionnaire de groupe du protocole $\texttt{Join}$. Cet oracle est utilisé par $\mathcal{A}$ jouant le rôle de l'utilisateur i contre le gestionnaire du groupe.
- $\mathcal{O}\texttt{Sign}(i, m, bsn)$ est un oracle utilisé par $\mathcal{A}$ pour demander une signature de l'utilisateur honnête i sur le message m et avec le marqueur bsn. $\mathcal{C}$ exécute alors $\texttt{Sign}(\mathsf{gsk}_i, m, bsn)$ et retourne σ à $\mathcal{A}$.
- $\mathcal{O}\texttt{Token}(bsn)$ est un oracle utilisé par $\mathcal{A}$ pour demander un jeton pour bsn. $\mathcal{C}$ exécute alors l'algorithme $\texttt{Token}(\mathsf{tsk}, bsn)$ et retourne le jeton τ correspondant.
- $\mathcal{O}\texttt{Open}(\sigma, m, bsn)$ est un oracle utilisé par $\mathcal{A}$ pour demander l'ouverture d'une signature σ sur m avec le marqueur bsn. $\mathcal{C}$ retourne alors le résultat de l'exécution de l'algorithme $\texttt{Open}(\sigma, m, bsn, \tau, \mathsf{osk}, \mathbf{Reg}, \mathbf{Sreg})$.

Traçabilité. La traçabilité d'un schéma AA-ODM stipule que seuls les utilisateurs admis par le gestionnaire de groupe puissent émettre des signatures et que toute signature valide doit permettre d'identifier son auteur via la procédure d'ouverture. Elle est formellement définie par le jeu de sécurité $\texttt{Exp}^{tra}_{\mathcal{A}}(1^k)$ décrit dans la figure 5.1.

$\texttt{Exp}^{tra}_{\mathcal{A}}(1^k)$

1. $p.p. \leftarrow \texttt{Setup}(1^k)$
2. $(\mathsf{isk}, \mathsf{ipk}, \mathsf{osk}, \mathsf{opk}, \mathsf{tsk}, \mathsf{tpk}) \leftarrow \texttt{Keygen}(p.p.)$
3. $(\sigma, m, bsn) \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Add}, \mathcal{O}\texttt{Corrupt}, \mathcal{O}\texttt{Sign}, \mathcal{O}\texttt{Join}_M}(\mathsf{osk}, \mathsf{tsk})$
4. Si $\texttt{Verify}(\sigma, m, bsn) = 0$, retourner 0.
5. $\tau \leftarrow \texttt{Token}(\mathsf{tsk}, bsn)$
6. Si $\texttt{Open}(\sigma, m, bsn, \tau, \mathsf{osk}, \mathbf{Reg}, \mathbf{Sreg}) = \perp$, retourner 1.
7. Sinon, retourner 0.

Figure 5.1 – Traçabilité

Soit $\texttt{Adv}^{tra}_{\mathcal{A}}(1^k) = \texttt{Pr}[\texttt{Exp}^{tra}_{\mathcal{A}}(1^k) = 1]$, la probabilité que l'adversaire $\mathcal{A}$ remporte ce jeu. Un schéma AA-ODM est *traçable* si cette valeur est négligeable pour tout adversaire probabiliste polynomial.

Non-Diffamation. Cette propriété spécifie qu'un utilisateur honnête ne peut être accusé à tort d'avoir émis une signature, même par une coalition regroupant toutes les autres entités du système.

Elle est formellement définie par le jeu de sécurité $\texttt{Exp}_{\mathcal{A}}^{ND}(1^k)$ décrit dans la figure 5.2.

$\texttt{Exp}_{\mathcal{A}}^{ND}(1^k)$

1. $p.p. \leftarrow \texttt{Setup}(1^k)$
2. $(\mathsf{isk}, \mathsf{ipk}, \mathsf{osk}, \mathsf{opk}, \mathsf{tsk}, \mathsf{tpk}) \leftarrow \texttt{Keygen}(p.p.)$
3. $(i, \sigma, m, bsn, \tau, \pi) \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Add}, \mathcal{O}\texttt{Corrupt}, \mathcal{O}\texttt{Join}_U, \mathcal{O}\texttt{Sign}}(\mathsf{isk}, \mathsf{osk}, \mathsf{tsk}, \mathbf{Sreg})$
4. Si i est corrompu, retourner 0.
5. Si σ a été retourné par l'oracle $\mathcal{O}\texttt{Sign}$, retourner 0.
6. Si $\texttt{Verify}(\sigma, m, bsn) = 0$, retourner 0.
7. Si $\texttt{Judge}(i, \sigma, m, bsn, \tau, \pi, \mathbf{Reg}) = 1$, retourner 1.
8. Sinon, retourner 0.

Figure 5.2 – Non-Diffamation

Soit $\texttt{Adv}_{\mathcal{A}}^{ND}(1^k) = \texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{ND}(1^k) = 1]$, la probabilité que l'adversaire $\mathcal{A}$ remporte ce jeu. Un schéma AA-ODM vérifie la propriété de non-diffamation si cette valeur est négligeable pour tout adversaire probabiliste polynomial.

Anonymat vis-à-vis de l'autorité de contrôle. Cette propriété précise que l'anonymat des utilisateurs honnêtes doit être préservé, même en présence d'une autorité de contrôle corrompue. Elle est définie par le jeu de sécurité $\texttt{Exp}_{\mathcal{A}}^{anoC\text{-}b^*}(1^k)$ décrit dans la figure 5.3.

$\texttt{Exp}_{\mathcal{A}}^{anoC\text{-}b^*}(1^k)$

1. $p.p. \leftarrow \texttt{Setup}(1^k)$
2. $(\mathsf{isk}, \mathsf{ipk}, \mathsf{osk}, \mathsf{opk}, \mathsf{tsk}, \mathsf{tpk}) \leftarrow \texttt{Keygen}(p.p.)$
3. $(i_0, i_1, m, bsn) \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Add}, \mathcal{O}\texttt{Corrupt}, \mathcal{O}\texttt{Sign}, \mathcal{O}\texttt{Join}_U, \mathcal{O}\texttt{Open}}(\mathsf{isk}, \mathsf{tsk})$
4. Si $\mathcal{O}\texttt{Join}_U(i_b)$ n'a pas été requis pour $b = 0$ ou $b = 1$, retourner 0.
5. $\sigma \leftarrow \texttt{Sign}(\mathsf{gsk}_{i_{b^*}}, m, bsn)$
6. $b' \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Add}, \mathcal{O}\texttt{Corrupt}, \mathcal{O}\texttt{Sign}, \mathcal{O}\texttt{Join}_U, \mathcal{O}\texttt{Open}}(\mathsf{isk}, \mathsf{tsk}, \sigma)$
7. Si i_0 ou i_1 est corrompu, retourner 0.
8. Si $\mathcal{O}\texttt{Sign}(i_b, ., bsn)$ a été requis pour $b = 0$ ou $b = 1$, retourner 0.
9. Si $\mathcal{O}\texttt{Open}(\sigma, m, bsn)$ a été requis, retourner 0.
10. Retourner $(b^* = b')$.

Figure 5.3 – Anonymat vis-à-vis de l'autorité de contrôle

Les restrictions des étapes 4 et 7 précisent simplement que les utilisateurs choisis par l'adversaire doivent être des membres honnêtes du groupe. Sans elles, la distinction serait rendue triviale. De même, l'adversaire ne peut évidemment pas faire une requête d'ouverture sur la signature dont il doit identifier l'auteur (étape 9). Finalement, la restriction de l'étape 8 empêche $\mathcal{A}$ d'obtenir une signature de l'un de ces utilisateurs avec le marqueur bsn. En effet, les propriétés des marqueurs rendraient alors la distinction possible en utilisant l'algorithme `Link`. Soit $\texttt{Adv}_{\mathcal{A}}^{anoC}(1^k) = \texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{anoC\text{-}1}(1^k) = 1] - \texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{anoC\text{-}0}(1^k) = 1]$. Un schéma AA-ODM est anonyme vis-à-vis de l'autorité de contrôle si cette valeur est négligeable pour tout adversaire probabiliste polynomial.

Anonymat vis-à-vis de l'autorité d'ouverture. Cette propriété, similaire à la précédente, considère cette fois-ci une autorité d'ouverture corrompue. Elle est définie par le jeu de sécurité $\mathtt{Exp}_{\mathcal{A}}^{anoO\text{-}b^*}(1^k)$ décrit dans la figure 5.4.

$\mathtt{Exp}_{\mathcal{A}}^{anoO\text{-}b^*}(1^k)$

1. $p.p. \leftarrow \mathtt{Setup}(1^k)$
2. $(\mathsf{isk}, \mathsf{ipk}, \mathsf{osk}, \mathsf{opk}, \mathsf{tsk}, \mathsf{tpk}) \leftarrow \mathtt{Keygen}(p.p.)$
3. $(i_0, i_1, m, bsn) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{Corrupt},\mathcal{O}\mathtt{Sign},\mathcal{O}\mathtt{Join}_U,\mathcal{O}\mathtt{Token}}(\mathsf{isk}, \mathsf{osk}, \mathbf{Sreg})$
4. Si $\mathcal{O}\mathtt{Join}_U(i_b)$ n'a pas été requis pour $b = 0$ ou $b = 1$, retourner 0.
5. $\sigma \leftarrow \mathtt{Sign}(\mathsf{gsk}_{i_{b^*}}, m, bsn)$
6. $b' \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{Corrupt},\mathcal{O}\mathtt{Sign},\mathcal{O}\mathtt{Join}_U,\mathcal{O}\mathtt{Token}}(\mathsf{isk}, \mathsf{osk}, \mathbf{Sreg}, \sigma)$
7. Si i_0 ou i_1 est corrompu, retourner 0.
8. Si $\mathcal{O}\mathtt{Sign}(i_b, ., bsn)$ a été requis pour $b = 0$ ou $b = 1$, retourner 0.
9. Si $\mathcal{O}\mathtt{Token}(bsn)$ a été requis, retourner 0.
10. Retourner $(b^* = b')$.

Figure 5.4 – Anonymat vis-à-vis de l'autorité d'ouverture

Les restrictions des étapes 4, 7, 8 et 9 sont imposées pour les mêmes raisons que précédemment. Soit $\mathtt{Adv}_{\mathcal{A}}^{anoC}(1^k) = \mathtt{Pr}[\mathtt{Exp}_{\mathcal{A}}^{anoO\text{-}1}(1^k) = 1] - \mathtt{Pr}[\mathtt{Exp}_{\mathcal{A}}^{anoO\text{-}0}(1^k) = 1]$. Un schéma AA-ODM est anonyme vis-à-vis de l'autorité d'ouverture si cette valeur est négligeable pour tout adversaire probabiliste polynomial.

Remarque 14. Les deux propriétés relatives à l'anonymat imposent à l'adversaire de distinguer des utilisateurs honnêtes. Cette restriction, identique à celle imposée par l'*anonymat désintéressé* des signatures de groupe [BS04], est ici indispensable. En effet, la connaissance de la clé secrète d'un utilisateur permet d'émettre des signatures pour n'importe quel marqueur, et donc en particulier le marqueur bsn choisi par l'adversaire. Par conséquent, l'algorithme `Link` pourrait être utilisé pour identifier l'émetteur de la signature σ.

5.2.3 Une Construction

Nous décrivons dans cette section un schéma AA-ODM que nous prouvons sûr dans le ROM sous des hypothèses standards.

L'idée de cette construction est d'imposer à l'utilisateur, lorsqu'il produit une signature, de fournir un élément K dépendant de manière déterministe de sa clé secrète, du marqueur et de la clé publique de l'autorité de contrôle. Ainsi, deux signatures produites avec la même clé et le même marqueur pourront immédiatement être reliées. Par ailleurs, cet élément permet également l'identification du signataire, mais à condition que les autorités de contrôle et d'ouverture collaborent.

Plus précisément, chaque utilisateur, lorsqu'il rejoint le groupe, génère un secret $y \in \mathbb{Z}_p$ et le fait certifier par le gestionnaire du groupe. Cette procédure assure également que l'utilisateur transmette l'élément $\widetilde{g}^y \in \mathbb{G}_2$ (pour un générateur $\widetilde{g}$ de $\mathbb{G}_2$) à l'autorité d'ouverture. Lorsqu'il produit une signature avec un marqueur bsn, l'utilisateur calcule l'élément $K \leftarrow e(J^y, T)$, où $T = \widetilde{g}^\delta$ est la clé publique de l'autorité de contrôle et $J \in \mathbb{G}_1$ est obtenu à partir de bsn. L'autorité de contrôle (dont la clé secrète est δ) et l'autorité d'ouverture (qui connaît $\widetilde{g}^y$) peuvent alors coopérer pour identifier les signatures émises par cet utilisateur. En effet, le jeton $\tau \leftarrow J^\delta$ permet à l'autorité d'ouverture de retrouver K en calculant $e(\tau, \widetilde{g}^y) = e(J, T)^y$.

Le point important est qu'aucune de ces deux entités ne peut identifier seule le secret y utilisé pour construire K. Pour l'autorité d'ouverture cela revient à résoudre le problème DBDH. Pour l'autorité de contrôle nous montrerons que le problème est au moins aussi dur que XDH.

De manière plus formelle, notre construction est définie par les algorithmes suivants.

- $\texttt{Setup}(1^k)$: soient $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires de type 3, g et h des générateurs de $\mathbb{G}_1$ et $\widetilde{g}$ un générateur de $\mathbb{G}_2$. L'algorithme sélectionne également les fonctions de hachage $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$ et $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ qui seront toutes deux modélisées comme des oracles aléatoires lors de l'analyse de sécurité. Les paramètres publics *p.p.* du système sont alors définis comme $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{H}, \mathcal{H}_1, \Sigma, \mathcal{H}_0)$ où Σ est un schéma de signature numérique et $\mathcal{H}_0 : \mathbb{G}_1 \to \mathcal{M}$ est une fonction de hachage à valeurs dans $\mathcal{M}$, l'espace des messages de Σ.
- $\texttt{Keygen}(p.p.)$: le gestionnaire du groupe initialise un registre **Reg** et sélectionne $\gamma \xleftarrow{\$} \mathbb{Z}_p$. Sa paire de clés $(\mathsf{isk}, \mathsf{ipk})$ est alors (γ, W), où $W \leftarrow \widetilde{g}^{\gamma}$. De même, l'autorité de contrôle définit $(\mathsf{tsk}, \mathsf{tpk}) \leftarrow (\delta, T)$ où $\delta \xleftarrow{\$} \mathbb{Z}_p$ et $T \leftarrow \widetilde{g}^{\delta}$. De son côté, l'autorité d'ouverture initialise le registre **Sreg** et génère une paire de clés $(\mathsf{osk}, \mathsf{opk})$ dépendant du contexte, comme nous l'expliquons dans la remarque 15. La clé publique gpk du groupe contient alors $(p.p., \mathsf{ipk}, \mathsf{tpk}, \mathsf{opk})$.
- $\texttt{UKeygen}()$: chaque utilisateur i génère une paire de clés $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \Sigma.\texttt{Keygen}()$. La clé pk_i est alors rendue publique.
- $\texttt{Join}$: l'objectif de cette procédure pour l'utilisateur est de générer un scalaire secret y qu'il fera certifier par le gestionnaire du groupe. Il doit également transmettre $\widetilde{g}^{y}$ à l'autorité d'ouverture pour permettre une identification des signatures qu'il produira.
  Pour des raisons de sécurité, il est nécessaire que y soit généré conjointement par l'utilisateur et le gestionnaire du groupe. Le protocole (inspiré de [DP06]) est alors le suivant.
  L'utilisateur génère $y_1 \xleftarrow{\$} \mathbb{Z}_p$ et calcule $C_1 \leftarrow g^{y_1}$ qu'il envoie au gestionnaire. Il prouve ensuite sa connaissance de y_1 à ce dernier qui, si la preuve est valide, lui retourne $y_2 \xleftarrow{\$} \mathbb{Z}_p$. L'utilisateur calcule alors $(C, \widetilde{C}) \leftarrow (g^{y_1+y_2}, \widetilde{g}^{y_1+y_2})$ et envoie C et $\mu_i \leftarrow \Sigma.\texttt{Sign}(\mathsf{sk}_i, \mathcal{H}_0(C))$ au gestionnaire. Celui-ci s'assure alors que :
  1. $C = C_1 \cdot g^{y_2}$;
  2. $\Sigma.\texttt{Verify}(\mathcal{H}_0(C), \mu_i, \mathsf{pk}_i) = 1$;
  3. $\widetilde{C}$ a été transmis à l'autorité d'ouverture (voir remarque ci-dessous) qui le conserve dans $\mathbf{Sreg}[i]$.

  Si tout est correct, il envoie à l'utilisateur $x \xleftarrow{\$} \mathbb{Z}_p$ et $A \leftarrow (h \cdot C)^{1/(\gamma+x)}$. La paire $(A, x) \in \mathbb{G}_1 \times \mathbb{Z}_p$ est alors (voir [DP06]) un certificat sur le secret $y \leftarrow y_1 + y_2$ dont la validité peut être testée à l'aide de l'équation :

$$e(A, W \cdot \widetilde{g}^{x}) = e(h, \widetilde{g}) \cdot e(g^{y}, \widetilde{g}).$$

  Si cette égalité est vérifiée, l'utilisateur définit $\mathsf{gsk}_i \leftarrow (y, A, x)$, tandis que le gestionnaire conserve (C, μ_i) dans $\mathbf{Reg}[i]$.
- $\texttt{Sign}(\mathsf{gsk}_i, m, bsn)$: pour signer un message m avec le marqueur bsn, l'utilisateur commence par calculer $J \leftarrow \mathcal{H}_1(bsn)$ et $K \leftarrow e(J^{y}, T)$. Il calcule ensuite une preuve de connaissance d'un certificat valide sur y qu'il transforme en signature de connaissance (voir section 3.3.3) sur m. Pour cela, il génère $k_x, k_y, k_d, k_z, d \xleftarrow{\$} \mathbb{Z}_p$ et calcule :
  1. $D \leftarrow A^{d}$;
  2. $R_1 \leftarrow e(D^{k_x} \cdot (g^{k_z} \cdot h^{k_d})^{-1}, \widetilde{g})$;
  3. $(R_2, R_3) \leftarrow (e(J^{k_y}, T), e(J^{k_z}, T) \cdot K^{-k_d})$;

4. $c \leftarrow \mathcal{H}(m, bsn, D, K, R_1, R_2, R_3)$;
5. $(s_x, s_y) \leftarrow (k_x + c \cdot x, k_y + c \cdot y)$;
6. $(s_d, s_z) \leftarrow (k_d + c \cdot d, k_z + c \cdot d \cdot y)$.

Il retourne alors la signature $\sigma = (D, K, c, s_x, s_y, s_d, s_z)$.

– $\texttt{Verify}(\sigma, m, bsn)$: la vérification d'une signature $\sigma = (D, K, c, s_x, s_y, s_d, s_z)$ consiste à s'assurer de la validité de la signature de connaissance précédente. Pour cela, l'algorithme calcule $R_1 \leftarrow e(D^{s_x} \cdot (g^{s_z} \cdot h^{s_d})^{-1}, \widetilde{g}) \cdot e(D, W)^c$, $R_2 \leftarrow e(\mathcal{H}_1(bsn)^{s_y}, T) \cdot K^{-c}$, $R_3 \leftarrow e(\mathcal{H}_1(bsn)^{s_z}, T) \cdot K^{-s_d}$, et vérifie que $c = \mathcal{H}(m, bsn, D, K, R_1, R_2, R_3)$. Il retourne 1 si c'est le cas et 0 sinon.

$\texttt{Link}(\sigma, m, \sigma', m', bsn)$: pour tester si les signatures σ et σ' portant le même marqueur ont été émises par le même utilisateur, l'algorithme commence par en vérifier la validité. Il compare ensuite l'élément $K \in \sigma$ avec $K' \in \sigma'$ et retourne 1 en cas d'égalité. Sinon, il retourne 0.

$\texttt{Token}(\mathsf{tsk}, bsn)$: pour émettre un jeton τ pour un marqueur bsn, l'autorité de contrôle calcule $\tau \leftarrow \mathcal{H}_1(bsn)^{\delta}$.

– $\texttt{Open}(\sigma, m, bsn, \tau, \mathsf{osk}, \mathbf{Reg}, \mathbf{Sreg})$: l'autorité d'ouverture commence par vérifier que τ est un jeton valide en testant si $e(\tau, \widetilde{g}) = e(\mathcal{H}_1(bsn), T)$ et que $\sigma = (D, K, c, s_x, s_y, s_d, s_z)$ est une signature valide sur m pour le marqueur bsn. Elle teste ensuite, pour chacun des éléments $\widetilde{g}^{y_i}$ conservés dans $\mathbf{Sreg}$, si $K = e(\tau, \widetilde{g}^{y_i})$ jusqu'à ce que l'égalité soit vérifiée (si aucune ne correspond, elle retourne $\perp$). Elle retourne alors l'indice i^* correspondant ainsi qu'une preuve de connaissance π de $\widetilde{g}^{y^*}$ vérifiant $K = e(\tau, \widetilde{g}^{y^*})$ et $e(C_{i^*}, \widetilde{g}) = e(g, \widetilde{g}^{y^*})$, où C_{i^*} est l'élément signé contenu dans $\mathbf{Reg}[i^*]$.

– $\texttt{Judge}(i, \sigma, m, bsn, \tau, \pi, \mathbf{Reg})$: pour vérifier la validité d'une ouverture, l'algorithme commence par tester celle du jeton $(e(\tau, \widetilde{g}) \stackrel{?}{=} (\mathcal{H}_1(bsn), T))$ et de la signature. Il récupère ensuite les éléments C_i et μ_i contenus dans $\mathbf{Reg}[i]$ et vérifie que :

1. π est valide, c'est-à-dire que π est une preuve de connaissance d'un élément $\widetilde{C}_i$ tel que $K = e(\tau, \widetilde{C}_i)$ et $e(C_i, \widetilde{g}) = e(g, \widetilde{C}_i)$;
2. μ_i est une signature valide sur C_i, c'est-à-dire que $\Sigma.\texttt{Verify}(\mathcal{H}_0(C_i), \mu_i, \mathsf{pk}_i) = 1$.

Si toutes ces conditions sont vérifiées, alors l'algorithme retourne 1. Sinon, il retourne 0.

Remarque 15. La capacité de l'autorité d'ouverture à lever l'anonymat repose sur la connaissance des valeurs $\widetilde{g}^{y_i}$ pour tous les membres i du groupe. Il est donc indispensable que ceux-ci lui révèlent cette information lors de leur adhésion. En pratique, il existe de très nombreuses façons de procéder. Une première solution consiste à faire intervenir l'autorité d'ouverture lors du protocole `Join`. Celle-ci peut alors confirmer la réception de l'élément $\widetilde{C} = \widetilde{g}^{y}$ tel que $e(C, \widetilde{g}) = e(g, \widetilde{C})$, par exemple en émettant une signature sur C pour une clé publique contenue dans opk. Cette solution, qui augmente le nombre d'interactions, peut être simplifiée si le gestionnaire de groupe et l'autorité d'ouverture sont une même entité, comme c'est le cas dans [BCN+10]. Malheureusement, cela rend la construction moins flexible dans le choix des entités. Une autre solution (similaire à celle décrite dans [BCLY08]), qui ne nécessite aucune interaction, serait que les utilisateurs chiffrent $\widetilde{C}$ sous la clé publique opk et prouve au gestionnaire de groupe (en utilisant par exemple les preuves Groth-Sahai) que l'élément chiffré vérifie bien $e(C, \widetilde{g}) = e(g, \widetilde{C})$. Ce dernier rajouterait alors le chiffré dans $\mathbf{Reg}[i]$, ce qui permettrait à l'autorité d'ouverture de récupérer $\widetilde{C}$.

Ainsi, le choix de la solution (et donc la définition des clés osk et opk) dépend en pratique du contexte et notamment des entités intervenant dans le système. Dans ce qui suit, nous supposerons donc simplement que l'autorité d'ouverture connait les éléments $\widetilde{g}^{y_i}$ pour tous les utilisateurs du système sans faire d'hypothèse sur la solution retenue.

Efficacité. Chaque signature σ de notre construction est donc constituée d'un élément de $\mathbb{G}_T$, d'un élément de $\mathbb{G}_1$ et de 5 scalaires. Cette taille relativement faible est notamment rendue

possible par le fait que notre construction ne nécessite pas de chiffrement. En implémentant nos groupes bilinéaires à l'aide des courbes Barreto-Naehrig [BN05], on obtient alors une signature de 577 octets. Nous reviendrons sur les moyens d'implémenter cette primitive sur des périphériques de faible puissance dans le prochain chapitre.

Les résultats relatifs à la sécurité de notre construction sont énoncés par le théorème suivant. Bien que dans le ROM, celle-ci repose sur des hypothèses standards.

Théorème 38. *Dans le modèle de l'oracle aléatoire, notre schéma est anonyme vis-à-vis de l'autorité de contrôle sous l'hypothèse* XDH *dans* $\mathbb{G}_1$*, anonyme vis-à-vis de l'autorité d'ouverture sous l'hypothèse* DBDH*, traçable sous l'hypothèse* $q-$SDH *et vérifie la propriété de non-diffamation sous l'hypothèse* SDL*, si* Σ *est un schéma de signature EUF-CMA et si* $\mathcal{H}_0$ *est une fonction de hachage résistante aux collisions.*

5.2.4 Étude de la Sécurité

Nous prouvons dans cette section le théorème 38 en étudiant séparément chaque propriété de sécurité. Certaines de ces preuves utiliseront le résultat suivant.

Lemme 39. *Le protocole décrit dans l'algorithme* `Sign` *est une signature de connaissance d'un scalaire* y *tel que* $K = e(\mathcal{H}_1(bsn), T)^y$ *ainsi que d'un certificat valide* $(A, x) \in \mathbb{G}_1 \times \mathbb{Z}_p$ *dessus.*

Preuve.
Complétude. Si le prouveur est honnête, alors :

$$\begin{aligned} e(D^{s_x} \cdot (g^{s_z} \cdot h^{s_d})^{-1}, \widetilde{g}) \cdot e(D, W)^c &= R_1 \cdot [e(D^x \cdot (g^{d \cdot y} \cdot h^d)^{-1}, \widetilde{g}) \cdot e(D, W)]^c \\ &= R_1 \cdot [e(A^x \cdot (g^y \cdot h)^{-1}, \widetilde{g}) \cdot e(A, W)]^{c \cdot d} \\ &= R_1 \cdot [e(A, W \cdot \widetilde{g}^x) \cdot e(g^y \cdot h, \widetilde{g})^{-1}]^{c \cdot d} \\ &= R_1. \end{aligned}$$

Le membre de droite se simplifie car (A, x) est un certificat valide sur y.

De même :

$$\begin{aligned} e(\mathcal{H}_1(bsn), T)^{s_y} \cdot K^{-c} &= R_2 \cdot e(\mathcal{H}_1(bsn), T)^{c \cdot y} \cdot K^{-c} \\ &= R_2 \\ e(\mathcal{H}_1(bsn), T)^{s_z} \cdot K^{-s_d} &= R_3 \cdot e(\mathcal{H}_1(bsn), T)^{c \cdot d \cdot y} \cdot K^{-c \cdot d} \\ &= R_3 \end{aligned}$$

ce qui prouve la complétude de la preuve.

Existence d'un extracteur. Supposons qu'un prouveur soit capable, pour une même mise en gage (R_1, R_2, R_3), de répondre (s_x, s_y, s_z, s_d) au challenge c et (s'_x, s'_y, s'_z, s'_d) au challenge $c' \neq c$. Soient $\overline{x} = \frac{s_x - s'_x}{c - c'}$, $\overline{y} = \frac{s_y - s'_y}{c - c'}$ et $\overline{d} = \frac{s_d - s'_d}{c - c'}$. Les deux réponses correspondant à la même mise en gage, on obtient :

$$\begin{aligned} e(\mathcal{H}_1(bsn), T)^{s_y} \cdot K^{-c} &= e(\mathcal{H}_1(bsn), T)^{s'_y} \cdot K^{-c'} \\ e(\mathcal{H}_1(bsn), T)^{s_z} \cdot K^{-s_d} &= e(\mathcal{H}_1(bsn), T)^{s'_y} \cdot K^{-s'_d} \end{aligned}$$

et donc $e(\mathcal{H}_1(bsn), T)^{\overline{y}} = K$ (première équation) ainsi que $e(\mathcal{H}_1(bsn), T)^{s_z - s'_z} = K^{s_d - s'_d}$ (deuxième équation). Par conséquent, on obtient la relation $s_z - s'_z = \overline{y}(s_d - s'_d)$.

De même, la mise en gage R_1 étant la même pour les deux réponses, on a :

$$\begin{aligned} e(D^{s_x} \cdot (g^{s_z} \cdot h^{s_d})^{-1}, \widetilde{g}) \cdot e(D, W)^c &= e(D^{s'_x} \cdot (g^{s'_z} \cdot h^{s'_d})^{-1}, \widetilde{g}) \cdot e(D, W)^{c'}, \\ e(D, \widetilde{g})^{s_x - s'_x} \cdot e(D, W)^{c-c'} &= e(g, \widetilde{g})^{s_z - s'_z} \cdot e(h, \widetilde{g})^{s_d - s'_d}, \\ e(D, \widetilde{g})^{s_x - s'_x} \cdot e(D, W)^{c-c'} &= e(g, \widetilde{g})^{\overline{y}(s_d - s'_d)} \cdot e(h, \widetilde{g})^{s_d - s'_d}, \\ e(D, \widetilde{g})^{\overline{x}} \cdot e(D, W) &= e(g, \widetilde{g})^{\overline{y} \cdot \overline{d}} \cdot e(h, \widetilde{g})^{\overline{x}} \text{ et donc} \\ e(D, W \cdot \widetilde{g}^{\overline{x}}) &= [e(g^{\overline{y}} \cdot h, \widetilde{g})]^d \end{aligned}$$

On distingue alors deux cas. Si $\overline{d} = 0$, on obtient $\overline{x} = -\gamma$, où γ est la clé secrète du gestionnaire de groupe. Il est alors possible de calculer un certificat sur $\overline{y}$. Sinon, l'équation précédente prouve que $(D^{\overline{d}^{-1}}, \overline{x})$ est un certificat valide sur $\overline{y}$.

Dans les tous cas, il est donc possible d'extraire $\overline{y}$ tel que $K = e(\mathcal{H}_1(bsn), T)^{\overline{y}}$ ainsi qu'un certificat valide sur ce scalaire.

Divulgation nulle. Le simulateur génère $D \xleftarrow{\$} \mathbb{G}_1$, $c \xleftarrow{\$} \mathbb{Z}_p$ et $s_x, s_y, s_d, s_z \xleftarrow{\$} \mathbb{Z}_p$ puis calcule $R_1 \leftarrow e(D^{s_x} \cdot (g^{s_z} \cdot h^{s_d})^{-1}, \widetilde{g}) \cdot e(D, W)^c$, $R_2 \leftarrow e(\mathcal{H}_1(bsn), T)^{s_y} \cdot K^{-c}$ et $R_3 \leftarrow e(\mathcal{H}_1(bsn), T)^{s_z} \cdot K^{-s_d}$. La trace $(D, R_1, R_2, R_3, c, s_x, s_y, s_z, s_d)$ est alors indistinguable de celle produite par un prouveur valide car d (tel que $D = A^d$) est choisi aléatoirement dans le protocole.

Preuve de l'anonymat vis-à-vis de l'autorité d'ouverture

L'objectif de l'adversaire $\mathcal{A}$ est de distinguer, parmi deux utilisateurs de son choix, lequel a émis une signature σ. Nous construisons une réduction $\mathcal{R}$ utilisant un tel adversaire pour résoudre le problème DBDH.

Soient $(g, g^a, g^b, g^c, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^b, \widetilde{g}^c)$ et $e(g, \widetilde{g})^z \in \mathbb{G}_T$ un challenge DBDH. Nous rappelons que l'objectif est de décider si $z = a \cdot b \cdot c$ ou si z est aléatoire.

La réduction $\mathcal{R}$ commence par choisir $n^* \in [1, q_j]$ et $k^* \in [1, q_h]$ où q_j est une borne sur le nombre de requêtes $\mathcal{O}\texttt{Join}_U$ et q_h est une borne sur le nombre de requêtes à l'oracle $\mathcal{O}\texttt{Hash}$ modélisant la fonction $\mathcal{H}_1$. Elle génère ensuite normalement les paramètres du système ainsi que les clés des différentes entités, à l'exception de celle de l'autorité de contrôle pour laquelle elle définit $T = \widetilde{g}^a$. Elle répond alors aux différents oracles de la manière suivante.

- $\mathcal{O}\texttt{Add}()$: $\mathcal{R}$ exécute l'algorithme $\texttt{UKeygen}$, conserve sk_i dans une liste $\mathcal{HK}$ et retourne la clé publique pk_i à $\mathcal{A}$.
- $\mathcal{O}\texttt{Join}_U(i)$: $\mathcal{R}$ exécute la partie utilisateur du protocole $\texttt{Join}$ s'il s'agit de la n-ième requête à cet oracle avec $n \neq n^*$. Sinon, elle procède comme si $C_1 = g^c$ en simulant la preuve de connaissance de c et en retournant $(g^c \cdot g^{y_2})$ à l'adversaire (y_2 est la valeur choisie par le gestionnaire de groupe contrôlé par $\mathcal{A}$). Ce dernier contrôlant également l'autorité d'ouverture, $\mathcal{R}$ lui transmet $(\widetilde{g}^c \cdot \widetilde{g}^{y_2})$. L'utilisateur impliqué dans ce dernier cas sera alors noté i^*. Sa clé secrète est implicitement définie comme étant $y^* = c + y_2$.
- $\mathcal{O}\texttt{Corrupt}(i)$: $\mathcal{R}$ retourne à $\mathcal{A}$ l'ensemble des valeurs secrètes de l'utilisateur i si $i \neq i^*$. Autrement, $\mathcal{R}$ arrête la simulation.
- $\mathcal{O}\texttt{Hash}(bsn)$: lorsque la k-ième nouvelle requête à cet oracle (c'est-à-dire une requête ne portant pas sur un bsn déjà soumis) est effectuée, $\mathcal{R}$ procède comme suit. Si $k \neq k^*$, alors elle choisit $u_k \xleftarrow{\$} \mathbb{Z}_p$, retourne g^{u_k} à $\mathcal{A}$ et conserve la paire (bsn, u_k) dans un registre. Sinon, elle choisit $v \xleftarrow{\$} \mathbb{Z}_p$ et retourne $(g^b)^v$ à $\mathcal{A}$ tout en conservant la paire (bsn, v). La valeur bsn impliquée dans ce cas est alors notée bsn^*.
  Si la requête porte sur un bsn déjà soumis, alors $\mathcal{R}$ retourne le même élément que la première fois en consultant son registre.
- $\mathcal{O}\texttt{Sign}(i, m, bsn)$: le comportement de $\mathcal{R}$ dépend de l'identité i et du marqueur bsn impliqués dans cette requête. Si $i \neq i^*$, alors elle exécute l'algorithme $\texttt{Sign}(\mathsf{gsk}_i, m, bsn)$ (car

elle connait gsk_i) et retourne la signature obtenue. Sinon, elle distingue deux cas. Si $bsn = bsn^*$, alors $\mathcal{R}$ arrête la simulation. Sinon, elle calcule $K = e(\mathcal{H}_1(bsn), T)^{y^*} \leftarrow e(g^c \cdot g^{y_2}, T)^u$ où u est le scalaire (connu de $\mathcal{R}$) tel que $\mathcal{O}\mathtt{Hash}(bsn) = g^u$. Elle simule ensuite la preuve de connaissance de y^* et de son certificat.

– $\mathcal{O}\mathtt{Token}(bsn)$: lorsque $\mathcal{A}$ demande un jeton sur bsn, $\mathcal{R}$ arrête la simulation si $bsn = bsn^*$ et retourne $(g^a)^u$ (où u est tel que $\mathcal{O}\mathtt{Hash}(bsn) = g^u$) sinon.

Lorsque $\mathcal{A}$ retourne (i_0, i_1, m, bsn), $\mathcal{R}$ arrête si $i^* \notin \{i_0, i_1\}$ ou si bsn a déjà été demandé lors de la k-ième requête (avec $k \neq k^*$) à l'oracle de hachage. Sinon, $\mathcal{R}$:

1. programme l'oracle $\mathcal{O}\mathtt{Hash}$ pour retourner $(g^b)^v$ pour un scalaire v aléatoire (ou passe à l'étape suivante si bsn a été demandé lors la k^*-requête à cet oracle) ;
2. calcule $K \leftarrow (e(g, \widetilde{g})^z)^v \cdot e(g^b, \widetilde{g}^a)^{v \cdot y_2}$;
3. simule la preuve de connaissance de y^* et de son certificat.

La simulation est alors parfaite si $z = a \cdot b \cdot c$. En effet, dans ce cas, $K = [e(g, \widetilde{g})^{a \cdot b \cdot v}]^{c+y_2} = e(\mathcal{H}_1(bsn), T)^{y^*}$. Autrement, K est un élément aléatoire de $\mathbb{G}_T$ et ne révèle aucune information sur i^*.

Pour que la simulation se poursuive jusqu'à la fin, il faut que i^* soit l'un des membres du groupe visés par $\mathcal{A}$ et que bsn ait été soumis lors de la k^*-ième requête à l'oracle $\mathcal{O}\mathtt{Hash}$ ou qu'il n'ait pas fait l'objet d'une requête avant la phase de « challenge ». Le premier événement survient avec une probabilité supérieure à $1/q_j$ tandis que le deuxième se produit avec une probabilité supérieure à $1/q_h$.

Preuve de l'anonymat vis-à-vis de l'autorité de contrôle

Cette preuve est assez similaire à la précédente à ceci près que l'adversaire $\mathcal{A}$ détient désormais les clés de l'autorité de contrôle et non celles de l'autorité d'ouverture. Nous montrons qu'un adversaire capable de remporter ce jeu de sécurité peut être utilisé pour résoudre le problème XDH dans $\mathbb{G}_1$.

Soient (g, g^a, g^b) et $g^z \in \mathbb{G}_1$ un challenge XDH. L'objectif est de décider si $z = a \cdot b$ ou si z est aléatoire.

La réduction $\mathcal{R}$ commence par choisir $n^* \in [1, q_j]$ et $k^* \in [1, q_h]$ où q_j est une borne sur le nombre de requêtes $\mathcal{O}\mathtt{Join}_U$ et q_h est une borne sur le nombre de requêtes à l'oracle $\mathcal{O}\mathtt{Hash}()$ précédemment défini. Elle génère ensuite normalement les paramètres du système ainsi que les clés des différentes entités (notamment γ et δ qu'elle envoie à $\mathcal{A}$). $\mathcal{R}$ répond aux différents oracles de la manière suivante.

– $\mathcal{O}\mathtt{Add}()$: $\mathcal{R}$ exécute l'algorithme $\mathtt{UKeygen}$, conserve sk_i dans une liste $\mathcal{HK}$ et retourne la clé publique pk_i à $\mathcal{A}$.

– $\mathcal{O}\mathtt{Join}_U(i)$: $\mathcal{R}$ exécute normalement la partie utilisateur de ce protocole s'il s'agit de la n-ième requête à cet oracle avec $n \neq n^*$. Sinon, elle procède comme si $C_1 = g^a$ en retournant $(g^a \cdot g^{y_2})$ et en simulant la preuve de connaissance de a. L'utilisateur impliqué dans ce dernier cas sera alors noté i^*. Sa clé secrète est implicitement définie comme étant $y^* = a + y_2$.

– $\mathcal{O}\mathtt{Corrupt}(i)$: $\mathcal{R}$ retourne à $\mathcal{A}$ l'ensemble des valeurs secrètes de l'utilisateur i si $i \neq i^*$. Autrement, $\mathcal{R}$ arrête la simulation.

– $\mathcal{O}\mathtt{Hash}(bsn)$: lorsque la k-ième nouvelle requête à cet oracle (c'est-à-dire une requête ne portant pas sur un bsn déjà soumis) est effectuée, $\mathcal{R}$ procède comme suit. Si $k \neq k^*$, alors elle choisit $u_k \xleftarrow{\$} \mathbb{Z}_p$, retourne g^{u_k} à $\mathcal{A}$ et conserve la paire (bsn, u_k) dans un registre. Sinon, elle choisit $v \xleftarrow{\$} \mathbb{Z}_p$ et retourne $(g^b)^v$ à $\mathcal{A}$ tout en conservant la paire (bsn, v). La valeur bsn impliquée dans ce cas est alors notée bsn^*.
Si la requête porte sur un bsn déjà soumis, alors $\mathcal{R}$ retourne le même élément que la première fois en consultant son registre.

– $\mathcal{O}\texttt{Sign}(i, m, bsn)$: le comportement de $\mathcal{R}$ dépend de l'identité i et du marqueur bsn impliqués dans cette requête. Si $i \neq i^*$, alors elle exécute l'algorithme $\texttt{Sign}(\texttt{gsk}_i, m, bsn)$ (car elle connait $\texttt{gsk}_i$) et retourne la signature obtenue. Sinon, elle distingue deux cas. Si $bsn = bsn^*$, alors $\mathcal{R}$ arrête la simulation. Sinon, elle calcule $K = e(\mathcal{H}_1(bsn), T)^{y^*} \leftarrow e(g^a \cdot g^{y_2}, T)^u$ où u est le scalaire (connu de $\mathcal{R}$) tel que $\mathcal{O}\texttt{Hash}(bsn) = g^u$. Elle simule ensuite la preuve de connaissance de y^* et de son certificat. Avant de retourner la signature σ à $\mathcal{A}$, elle en conserve une copie dans un registre **SignReg**.
– $\mathcal{O}\texttt{Open}(\sigma, m, bsn)$: lorsque $\mathcal{A}$ demande l'ouverture d'une signature valide σ, $\mathcal{R}$ commence par regarder si σ figure dans le registre **SignReg** auquel cas elle retourne i^* et simule la preuve de connaissance de g^{y^*}. Sinon, elle calcule le jeton τ correspondant à bsn, exécute l'algorithme $\texttt{Open}(\sigma, m, bsn, \tau, \texttt{osk}, \mathbf{Reg}, \mathbf{Sreg})$ et retourne le résultat à $\mathcal{A}$.

Lorsque $\mathcal{A}$ retourne (i_0, i_1, m, bsn), $\mathcal{R}$ arrête si $i^* \notin \{i_0, i_1\}$ ou si bsn a déjà été demandé lors de la k-ième requête (avec $k \neq k^*$) à l'oracle de hachage. Sinon, $\mathcal{R}$:

1. programme l'oracle $\mathcal{O}\texttt{Hash}$ pour retourner $(g^b)^v$ pour un scalaire v aléatoire (ou passe à l'étape suivante si bsn a été demandé lors la k^*-requête à cet oracle) ;
2. calcule $K \leftarrow e(g^z, T)^v \cdot e(g^b, T)^{v \cdot y_2}$;
3. simule la preuve de connaissance de y^* et de son certificat.

La simulation est alors parfaite si $z = a \cdot b$. En effet, dans ce cas, $K = [e(g, T)^{=b \cdot v}]^{a+y_2} = e(\mathcal{H}_1(bsn), T)^{y^*}$. Autrement, K est un élément aléatoire de $\mathbb{G}_T$ et ne révèle aucune information sur i^*.

Pour que la simulation se poursuive jusqu'à la fin, il faut que i^* soit l'un des membres du groupe visés par $\mathcal{A}$ et que bsn ait été soumis lors de la k^*-ième requête à l'oracle $\mathcal{O}\texttt{Hash}$ ou qu'il n'ait pas fait l'objet d'une requête avant la phase de « challenge ». Le premier événement survient avec une probabilité supérieure à $1/q_j$ tandis que le deuxième se produit avec une probabilité supérieure à $1/q_h$.

Remarque 16.
– La propriété d'anonymat vis-à-vis de l'autorité de contrôle repose donc sur l'hypothèse XDH qui est considérée comme raisonnable pour les groupes bilinéaires de type 3. On peut cependant remarquer que cette propriété peut reposer sur une hypothèse plus faible où le challenge serait constitué de $(g, g^a, g^b, \widetilde{g}) \in \mathbb{G}_1^3 \times \mathbb{G}_2$ et de $e(g, \widetilde{g})^z \in \mathbb{G}_T$. L'élément K retourné par $\mathcal{R}$ lors de la phase de « challenge » serait alors $(e(g, \widetilde{g})^z)^{v \cdot \delta} \cdot e(g^b, T)^{v \cdot y_2}$.
– Dans la simulation précédente, $\mathcal{R}$ est capable de répondre correctement à toutes les requêtes à l'oracle $\mathcal{O}\texttt{Open}$ à condition que $\mathcal{A}$ ne puisse pas contrefaire la signature de i^*. Dans le cas contraire, $\mathcal{A}$ pourrait soumettre une signature valide σ ne figurant pas dans le registre **SignReg** et que $\mathcal{R}$ ne pourrait identifier car elle ne connait pas $\widetilde{g}^{y^*}$. Cependant, un adversaire capable de produire une telle contrefaçon avec une probabilité non-négligeable contredirait la propriété de non-diffamation. Nous prouvons ci-dessous qu'un tel adversaire n'existe pas sous les hypothèses mentionnées dans le théorème 38.

Preuve de la propriété de non-diffamation

L'objectif de l'adversaire $\mathcal{A}$ est de produire une preuve π accusant un utilisateur honnête i d'avoir produit une signature σ sur m pour le marqueur bsn. Cette preuve doit pouvoir être publiquement vérifiée grâce à l'algorithme Judge et au contenu du registre public **Reg**. Soit (C_i, μ_i) la paire fournie par $\mathcal{R}$ lors de la requête $\mathcal{O}\texttt{Join}_U(i)$. Nous distinguons les trois cas suivants.

– Cas 1 : $\mathbf{Reg}[i]$ contient une paire (C_i', μ_i) avec $C_i' \neq C_i$.
– Cas 2 : $\mathbf{Reg}[i]$ contient une paire (C_i', μ_i') avec $C_i' \neq C_i$ et $\mu_i' \neq \mu_i$.
– Cas 3 : $\mathbf{Reg}[i]$ contient C_i.

La preuve π établit le lien entre l'élément K contenu dans la signature σ et l'élément C'_i contenu dans $\mathbf{Reg}[i]$. Pour que la preuve soit jugée valide par l'algorithme `Judge`, il faut que $\mathbf{Reg}[i]$ contienne également une signature valide sur cet élément pour la clé publique pk_i de l'utilisateur. Le cas 1 signifie que l'adversaire a remplacé l'élément C_i original par un élément différent C'_i pour lequel la signature μ_i reste valide. Il implique cependant une collision $\mathcal{H}_0(C_i) = \mathcal{H}_0(C'_i)$ pour la fonction de hachage $\mathcal{H}_0$ et ne survient donc qu'avec une probabilité négligeable. Le cas 2 signifie que l'adversaire a été capable de produire une signature μ'_i sur un nouvel élément C'_i. Là encore, il est très improbable en raison de la propriété EUF-CMA du schéma de signature Σ. Le cas 3 est traité par le lemme suivant.

Lemme 40. *Un adversaire $\mathcal{A}$ produisant des attaques correspondant au cas 3 peut être converti en un adversaire contre l'hypothèse* SDL.

Preuve. Soit $(g, g^a, \widetilde{g}, \widetilde{g}^a) \in \mathbb{G}_1 \times \mathbb{G}_2$ un challenge SDL. $\mathcal{R}$ exécute les algorithmes `Setup` et `Keygen` et envoie les clés secrètes des différentes entités à $\mathcal{A}$. Il sélectionne ensuite un entier $n^* \in [1, q_j]$ où q_j est une borne sur le nombre de requêtes $\mathcal{O}\mathtt{Join}_U$ et répond aux différents oracles comme suit.

- $\mathcal{O}\mathtt{Add}()$: $\mathcal{R}$ exécute l'algorithme `UKeygen`, conserve sk_i dans une liste $\mathcal{HK}$ et retourne la clé publique pk_i à $\mathcal{A}$.
- $\mathcal{O}\mathtt{Join}_U(i)$: $\mathcal{R}$ exécute normalement la partie utilisateur de ce protocole s'il s'agit de la n-ième requête à cet oracle avec $n \neq n^*$. Sinon, elle procède comme si $C_1 = g^a$ en simulant la preuve de connaissance de a et en retournant $(g^a \cdot g^{y_2})$ à l'adversaire (y_2 est la valeur choisie par le gestionnaire de groupe contrôlé par $\mathcal{A}$). Ce dernier contrôlant également l'autorité d'ouverture, $\mathcal{R}$ lui transmet $(\widetilde{g}^a \cdot \widetilde{g}^{y_2})$. L'utilisateur impliqué dans ce dernier cas sera alors noté i^*. Sa clé secrète est implicitement définie comme étant $y^* = a + y_2$.
- $\mathcal{O}\mathtt{Corrupt}(i)$: $\mathcal{R}$ retourne à $\mathcal{A}$ l'ensemble des valeurs secrètes de l'utilisateur i si $i \neq i^*$. Autrement, $\mathcal{R}$ arrête la simulation.
- $\mathcal{O}\mathtt{Sign}(i, m, bsn)$: le comportement de $\mathcal{R}$ dépend de l'identité i impliquée dans cette requête. Si $i \neq i^*$, alors elle exécute l'algorithme $\mathtt{Sign}(\mathsf{gsk}_i, m, bsn)$ (car elle connait gsk_i) et retourne la signature obtenue. Sinon, elle calcule $K \leftarrow e(\mathcal{H}_1(bsn), \widetilde{g}^a \cdot \widetilde{g}^{y_2})^\delta$ et simule la signature de connaissance de $y^* = a + y_2$ et de son certificat.

À la fin de la simulation, $\mathcal{A}$ retourne une signature valide σ ainsi qu'une preuve π que celle-ci a été produite par un utilisateur i. Si $i \neq i^*$, alors $\mathcal{R}$ arrête. Sinon, étant dans le cas 3, l'adversaire n'a pas modifié le registre $\mathbf{Reg}[i^*]$ qui contient par conséquent l'élément $C_{i^*} = g^{a+y_2}$ que $\mathcal{R}$ a envoyé lors de la requête $\mathcal{O}\mathtt{Join}_U$. La preuve π assure alors que l'élément K contenu dans σ est égal à $e(\mathcal{H}_1(bsn), T)^{a+y_2}$ et donc que la signature contient une preuve de connaissance de $a + y_2$. Ce scalaire peut alors être extrait, ce qui permet à $\mathcal{R}$ de retrouver a car elle connait y_2.

$\mathcal{R}$ peut donc utiliser $\mathcal{A}$ contre l'hypothèse SDL à condition d'avoir correctement deviné l'identité de l'utilisateur « piégé » par $\mathcal{A}$, ce qui arrive avec probabilité $1/q_j$.

Preuve de la traçabilité

Un adversaire remportant le jeu de la traçabilité est capable de produire une signature valide σ dont l'émetteur ne peut être identifié. Cela signifie que σ contient un élément $K = e(\mathcal{H}_1(bsn), T)^y$ pour un scalaire y qui n'a jamais été certifié lors d'une requête $\mathcal{O}\mathtt{Join}_M$. En effet, dans le cas contraire, le registre **Sreg** contiendrait l'élément $\widetilde{g}^y$ et l'algorithme `Open` n'aurait donc pas retourné $\perp$. Le lemme 39 garantit alors qu'il est possible d'extraire de σ un certificat (A, x) sur y vérifiant :

$$e(A, W \cdot \widetilde{g}^x) = e(h, \widetilde{g}) \cdot e(g^y, \widetilde{g})$$

La section 4.5 de l'article [DP06] montre alors comment utiliser ce nouveau certificat pour résoudre le problème $q_j - \mathsf{SDH}$, où q_j est une borne sur le nombre de requêtes $\mathcal{O}\mathtt{Join}_M$ soumises par $\mathcal{A}$.

Troisième partie

Optimisation de Protocoles Cryptographiques

Chapitre 6

Délégation d'Algorithmes Cryptographiques

Sommaire

Nous présentons dans ce chapitre certaines techniques permettant d'implémenter des protocoles cryptographiques complexes sur des périphériques de faible puissance. Nous nous intéressons plus particulièrement à la délégation d'algorithmes cryptographiques que nous appliquons à des protocoles étudiés dans les chapitres précédents. Les résultats présentés ici sont issus des articles *Toward Generic Method for Server-Aided Cryptography* [CCD+13], publié à la conférence ICICS 2013, et *Efficient Delegation of Zero-Knowledge Proofs of Knowledge in a Pairing-Friendly Setting* [CPS14], publié à la conférence PKC 2014.

6.1 Externalisation de Calculs

6.1.1 Contexte

Les exemples du paiement anonyme (chapitre 4) et de l'authentification anonyme (chapitre 5) illustrent parfaitement la capacité de la cryptographie à s'adapter aux problèmes de sécurité des différents cas d'usage. Elle permet, en effet, de répondre à des exigences a priori contradictoires, telles que l'anonymat et l'authentification. Cependant, cette flexibilité n'est pas gratuite, chaque nouvelle fonctionnalité rajoutée à un protocole a tendance à le rendre plus complexe.

Le problème de l'efficacité peut sembler anodin lorsqu'un protocole est implémenté sur un ordinateur. En effet, les différents schémas que nous avons décrits ne nécessitent que quelques calculs « classiques », tels que des exponentiations ou des couplages, qu'un ordinateur peut exécuter en quelques millisecondes (voir par exemple [BGM+10]). Malheureusement, bon nombre de protocoles ne deviennent intéressants qu'une fois implémentés sur des périphériques mobiles. Cela est d'autant plus vrai pour les cas du paiement de proximité ou des transports publics où un

utilisateur ne va certainement pas pouvoir s'authentifier à l'aide d'un ordinateur. Il est donc indispensable que cela se fasse à l'aide d'un périphérique aisément transportable, en général moins puissant.

L'évolution de la puissance des téléphones mobiles pourrait apporter une réponse mais malheureusement ceux-ci n'offrent pas les garanties de sécurité suffisantes pour stocker des données aussi sensibles que des clés secrètes ou des informations bancaires. Ces données doivent être conservées sur un périphérique sécurisé, par exemple la carte SIM, nettement moins puissante.

Concevoir des protocoles plus performants est l'une des solutions apportées aux problèmes de l'implémentation de primitives complexes sur des périphériques de faible puissance. À ce titre, les travaux que nous avons présentés dans les deux chapitres précédents, mais également ceux de Bichsel *et al.* [BCN+10] sur les signatures de groupe ou ceux de Bernhard *et al.* [BFG+13] sur les attestations anonymes, permettent des constructions relativement peu coûteuses. Malheureusement, cela n'est pas toujours suffisant pour les implémenter sur une carte à puce.

Le premier problème est celui du temps d'exécution. En effet, les spécifications des différents cas d'usage imposent des contraintes de temps extrêmement fortes. Par exemple, dans le cas du transport public, l'authentification d'un utilisateur, à savoir l'émission de la signature par la carte et sa vérification par la borne, doit pouvoir être effectuée en moins de 300 ms [GSM12]. À titre de comparaison, une exponentiation dans le groupe $\mathbb{G}_1$ d'une courbe Barreto-Naehrig [BN05] nécessite environ 50 ms sur une carte SIM NFC récente. Il semble donc, a priori, impossible de respecter les exigences des différents cas d'usage.

Le deuxième problème est que les cartes à puce ne supportent généralement pas toutes les opérations intervenant dans un protocole. C'est notamment le cas du couplage sur courbes elliptiques dont l'implémentation sur carte reste limitée à quelques prototypes de recherche. Là encore, cela semble exclure plusieurs protocoles, par exemple celui décrit dans le chapitre 5.

Cependant, les périphériques informatiques sont aujourd'hui rarement isolés. Si nous reprenons l'exemple de la carte SIM, celle-ci est embarquée dans un téléphone portable dont les performances n'ont cessé de croître, concurrençant désormais celles d'un ordinateur. Il est donc naturel de se demander si la carte SIM ne peut pas tirer parti de cette puissance de calcul disponible. Malheureusement, elle ne peut pas simplement déléguer l'intégralité du protocole au téléphone. Cela reviendrait à lui confier les données secrètes de l'utilisateur, ce que nous avions exclu pour des raisons de sécurité. La délégation d'algorithmes cryptographiques nécessite donc une étude plus poussée afin de satisfaire des exigences de sécurité dépendant du contexte.

Nous considérons dans cette section le problème de la répartition des différentes opérations composant un algorithme entre une entité de confiance mais de faible puissance (*e.g.* une carte SIM) et une autre entité beaucoup plus puissante (*e.g.* un téléphone) mais potentiellement corrompue (nous préciserons cette notion pour quelques exemples ci-dessous). Nous étudierons le problème de la délégation des opérations elles-mêmes dans le prochain chapitre.

Il est à noter qu'il existe des techniques génériques de délégation, en particulier celles basées sur le chiffrement totalement homomorphe [Gen09]. Cependant, celles-ci n'offrent en général pas la même efficacité que les protocoles de délégation conçus spécifiquement pour un algorithme ou une opération. Concernant le chiffrement totalement homomorphe, même si de nombreux travaux (*e.g.* [vDGHV10, CCK+13, CLT14b]) ont permis d'en améliorer significativement les performances, il semble difficile d'envisager une implémentation sur carte à puce dans un futur proche.

6.1.2 Quelques Exemples

La première étape de la délégation d'un algorithme consiste à définir précisément les propriétés de sécurité attendues vis-à-vis du délégataire. Celles-ci dépendent de la primitive et du contexte et doivent donc être définies au cas par cas. Nous nous intéresserons ici à la délégation du protocole `Spend` du schéma de monnaie électronique présenté dans la section 4.3 et du protocole `Sign` du schéma AA-ODM que nous avons présenté dans le chapitre précédent. En effet,

ces deux protocoles sont ceux qui doivent satisfaire en pratique les contraintes de temps les plus fortes. Pour plus de simplicité nous désignerons l'entité déléguant une partie des calculs par « la carte SIM » et le délégataire par « le téléphone ». Bien évidemment, les techniques que nous décrivons ici s'appliquent à n'importe quelles autres entités, telles qu'un ordinateur et un serveur par exemple.

Le problème de la délégation de protocoles cryptographiques anonymes a déjà été étudié dans le cas de la signature de groupe par Maitland et Boyd [MB02] et Canard *et al.* [CCdMP10]. Les schémas d'attestations anonymes [BCC04, BFG+13] le considèrent systématiquement en répartissant les calculs de l'algorithme `Sign` entre une entité, appelée *TPM* (correspondant à notre carte SIM), et une autre, appelée *Host* (correspondant au téléphone). L'efficacité des protocoles résultant de cette délégation repose sur l'affaiblissement de certaines propriétés de sécurité vis-à-vis du téléphone. Nous suivrons la même idée en considérant que le téléphone n'est pas corrompu lors des différents jeux de sécurité définissant l'anonymat des primitives. Concrètement, cela signifie que l'anonymat vis-à-vis du téléphone n'est pas exigé (car il dispose de toute façon d'autres moyens d'identifier son utilisateur) mais que les données auxquelles il a accès ne doivent pas aider un adversaire à casser les autres propriétés de sécurité. Dans le cas d'un schéma de monnaie électronique divisible, cela garantit que le téléphone seul (sans la carte SIM) ne peut pas effectuer de paiement. Dans le cas d'un schéma AA-ODM, cela garantit que le téléphone seul ne peut pas émettre de signature.

De manière générale, la délégation de calcul soulève un autre problème que celui de la confidentialité des données impliquées dans l'exécution de l'algorithme. En effet, un téléphone malhonnête pourrait également retourner des valeurs incorrectes. Cependant, dans les deux cas que nous considérons ici, cela entrainerait simplement la production d'une dépense (resp. signature) invalide qui serait rejetée par le marchand (resp. la borne). Se prémunir contre ce type d'attaques n'est donc pas réellement pertinent car un téléphone corrompu pourra toujours empêcher l'utilisation du service.

Délégation de la dépense d'un schéma de monnaie électronique divisible

Afin d'obtenir les meilleures performances, nous considérons ici une instanciation dans le ROM de la version non-corrélable du schéma de la section 4.3. Le schéma de signature Σ_1 sera implémenté avec la construction de [BFG+13] qui est une variante des signatures Camenisch-Lysyanskaya que nous avons présentées dans la section 3.2.2. Celle-ci permet d'éviter de manipuler des éléments de $\mathbb{G}_T$ dans la preuve de connaissance, mais au prix d'une taille de signature un peu plus importante.

Pour une meilleure compréhension de ce qui suit, nous commençons par décrire le schéma de signature Blind-LRSW de [BFG+13] ainsi que le protocole `Spend` obtenu dans ce cas-là.

Signatures Blind-LRSW.

- $\mathtt{Setup}(1^k)$: cet algorithme retourne les paramètres *p.p.* contenant la description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ de groupes bilinéaires ainsi qu'un générateur g (resp. $\widetilde{g}$) de $\mathbb{G}_1$ (resp. $\mathbb{G}_2$).
- $\mathtt{Keygen}(p.p.)$: cet algorithme retourne $\mathsf{sk} = (x, y) \xleftarrow{\$} \mathbb{Z}_p^2$ et $\mathsf{pk} = (\widetilde{X}, \widetilde{Y}) \leftarrow (\widetilde{g}^x, \widetilde{g}^y)$.
- $\mathtt{Sign}(\mathsf{sk}, g^m)$: cet algorithme prend en entrée g^m pour un message $m \in \mathbb{Z}_p$ et retourne $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4) \leftarrow (g^r, g^{r \cdot y}, g^{r(x + m \cdot x \cdot y)}, g^{r \cdot y \cdot m})$ pour un scalaire r aléatoire.
- $\mathtt{Verify}(\mathsf{pk}, m, \sigma)$ pour vérifier que σ est une signature valide sur m, cet algorithme teste si les égalités $\sigma_4 = \sigma_2^m$, $e(\sigma_1, \widetilde{Y}) = e(\sigma_2, \widetilde{g})$ et $e(\sigma_3, \widetilde{g}) = e(\sigma_1 \cdot \sigma_4, \widetilde{X})$ sont vérifiées. Il retourne 1 si c'est le cas et 0 sinon.

Lors d'un retrait, ce schéma va permettre à la banque de délivrer un certificat $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ sur le secret x choisi par l'utilisateur. La partie utilisateur du protocole de dépense (`Spend`) est alors le suivant.

$\mathtt{Spend}(\mathcal{U}(C, \mathsf{bpk}, \mathsf{mpk}, 2^\ell), \mathcal{M}(\mathsf{msk}, \mathsf{bpk}, 2^\ell))$: pour dépenser une valeur 2^ℓ, l'utilisateur

1. sélectionne un sommet s non dépensé de niveau $n - \ell$;
2. calcule $t_s \leftarrow g_s^x$;
3. régénère σ en calculant $\sigma' = (\sigma'_1, \sigma'_2, \sigma'_3, \sigma'_4) = (\sigma_1^t, \sigma_2^t, \sigma_3^t, \sigma_4^t)$ pour un scalaire t aléatoire ;
4. calcule $(R_1, R_2) \leftarrow (g_s^k, (\sigma'_2)^k)$;
5. calcule $c \leftarrow \mathcal{H}(\sigma', t_s, R_1, R_2, info)$ et $z \leftarrow k + c \cdot x$;
6. retourne $(Z, \pi) \leftarrow (t_s, (\sigma', c, z))$.

Nous rappelons qu'il n'est pas nécessaire de calculer l'élément v_s (voir section 4.3) ou de masquer l'élément g_s utilisé car nous ne visons ici que la propriété de non-corrélation. Une dépense implique donc le transfert de 5 éléments de $\mathbb{G}_1$ et de 2 scalaires. Elle nécessite essentiellement d'effectuer 7 exponentiations dans $\mathbb{G}_1$, les autres opérations ayant un coût négligeable en comparaison. Transférer certains de ces calculs au téléphone est une solution pour améliorer le temps d'exécution du protocole mais il faut tout d'abord déterminer quelles sont les données auxquelles il peut avoir accès.

Intuitivement, l'idée est de s'assurer que la corruption du téléphone ne permette pas à l'adversaire d'accéder à des données sensibles grâce auxquelles il pourrait casser une propriété de sécurité. Formellement, cela se traduit par l'ajout d'un oracle $\mathcal{O}\texttt{CoopSpend}(\texttt{upk}, v)$ dans les jeux concernés. Cet oracle, exécutant la partie SIM du protocole `Spend`, est utilisé par l'adversaire jouant le rôle du téléphone.

Comme nous l'avons expliqué, cette corruption n'est pas possible dans le jeu de l'anonymat mais seulement dans les autres, à savoir ceux de la traçabilité et de la propriété de non-diffamation.

En ce qui concerne la traçabilité, on peut noter que l'adversaire peut corrompre tous les utilisateurs du système. Par conséquent, la corruption du téléphone ne lui apporte aucun pouvoir supplémentaire, même si ce dernier contient $C = (\sigma, x)$, l'ensemble des données relatives à la pièce.

La propriété de non-diffamation impose, quant à elle, une restriction sur les requêtes $\mathcal{O}\texttt{Corrupt}$. L'utilisateur visé par l'adversaire ne doit en effet pas avoir été corrompu. Il est d'ailleurs assez simple de montrer que la connaissance de C suffit à casser cette propriété. Par conséquent, le téléphone ne peut y avoir accès. Cependant, l'adversaire connait déjà le certificat σ de chaque pièce car il contrôle la banque lors de ce jeu. Le conserver sur le téléphone ne pose donc, a priori, pas de problème, même en cas de corruption de ce dernier.

Dans ce qui suit nous supposerons donc que le téléphone a accès au certificat σ de la pièce mais pas à la valeur x qui ne sera connue que de la carte SIM. Il est alors possible de répartir les différentes opérations de la procédure de dépense de la manière décrite dans la figure 6.1.

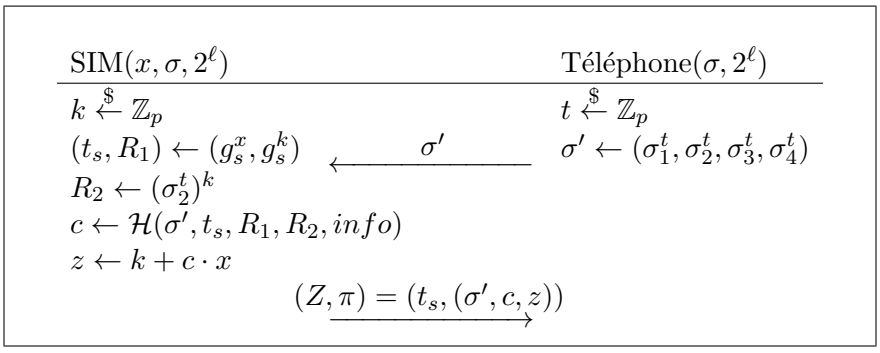


Figure 6.1 – Délégation de la partie utilisateur du protocole `Spend`.

Cette version coopérative du protocole ne nécessite donc plus que 3 exponentiations de la part de la carte SIM. Il est assez simple de prouver que les propriétés de traçabilité et de non-diffamation restent satisfaites, même en présence de l'oracle $\mathcal{O}\texttt{CoopSpend}$. En effet, dans les deux

cas, la réduction $\mathcal{R}$ construite dans les preuves de sécurité connait les certificats de toutes les pièces des utilisateurs honnêtes du système. Par conséquent, elle est capable de gérer toutes les requêtes à ce nouvel oracle.

Délégation de la signature du schéma AA-ODM

Les propriétés de traçabilité et de non-diffamation d'un schéma AA-ODM présentent bon nombre de similarités avec celles d'un schéma de monnaie électronique divisible. Il est donc possible d'appliquer le raisonnement précédent et de confier certaines des données constituant la clé gsk de l'utilisateur au téléphone. Plus précisément, cette clé contient un scalaire y connu seulement de l'utilisateur et un certificat (A, x) (sur y) délivré par le gestionnaire de groupe. Là encore, ce certificat est déjà connu de l'adversaire lors du jeu définissant la traçabilité et lors de celui définissant la non-diffamation. Par conséquent, nous autoriserons le téléphone à y avoir accès et à faire les calculs en dépendant.

Afin de modéliser l'exécution coopérative de l'algorithme `Sign`, nous rajoutons un oracle $\mathcal{O}\texttt{CoopSign}(i, m, bsn)$ exécutant la partie SIM de celui-ci. Le protocole résultant de cette délégation satisfera donc les propriétés de traçabilité et de non-diffamation si la réduction construite dans les preuves de la section 5.2.4 est capable de simuler correctement les réponses aux requêtes correspondantes.

Nous décrivons dans la figure 6.2 une répartition possible des calculs de l'algorithme `Sign` entre une carte SIM connaissant $\mathsf{gsk} = (y, A, x)$ et le téléphone connaissant (A, x).

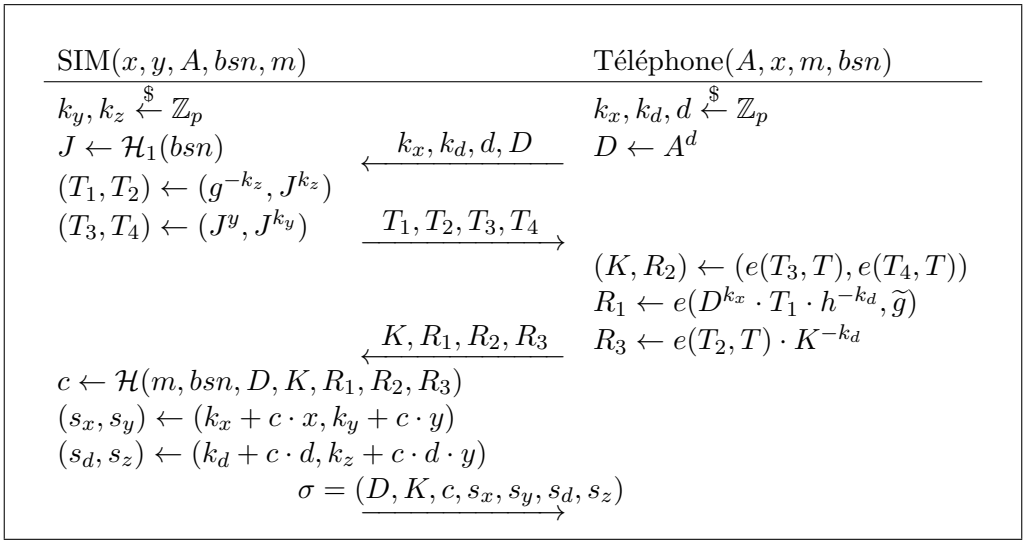


Figure 6.2 – Délégation de l'algorithme `Sign` de la section 5.2.3

Cette exécution coopérative permet à la carte SIM de n'effectuer plus que 4 exponentiations dans $\mathbb{G}_1$, contre 7 auparavant. Par ailleurs, celle-ci n'a plus besoin de calculer les 4 couplages et l'exponentiation dans $\mathbb{G}_T$ qu'elle devait effectuer initialement. Si la réduction du jeu de la traçabilité peut facilement simuler les réponses aux requêtes $\mathcal{O}\texttt{CoopSign}$ car elle connait les clés gsk de tous les utilisateurs honnêtes, le cas de la non-diffamation nécessite un peu plus de détails. En effet, rappelons que la réduction $\mathcal{R}$ de cette preuve ne connait que la paire $(g^y, \widetilde{g}^y)$ ainsi que le certificat (A, x). Pour répondre à une requête $\mathcal{O}\texttt{CoopSign}$ portant sur un message m et un marqueur bsn, elle commence par envoyer le certificat (A, x) à $\mathcal{A}$ et par générer $c, s_y, s_z \xleftarrow{\$} \mathbb{Z}_p$. Elle programme ensuite la fonction $\mathcal{H}_1$ afin de connaître le scalaire u tel que $J = \mathcal{H}_1(bsn) = g^u$. Lorsque l'adversaire lui envoie $(k_x, k_d, d, D) \in \mathbb{Z}_p^3 \times \mathbb{G}_1$, elle lui retourne $T_1 \leftarrow g^{-s_z} \cdot (g^y)^{d \cdot c}$,

$T_2 \leftarrow T_1^{-u}$, $T_3 \leftarrow (g^y)^u$ et $T_4 \leftarrow g^{s_y \cdot u} \cdot (g^y)^{-c \cdot u}$. Il ne reste alors plus qu'à programmer $\mathcal{H}$ pour retourner c et à envoyer les scalaires s_x, s_y, s_z et s_d. La simulation est alors parfaite.

Par conséquent, le protocole décrit dans la figure ci-dessus garantit les propriétés de traçabilité et de non-diffamation, même en cas de corruption du téléphone.

6.2 Délégation de Preuve de Connaissance

Les exemples que nous avons décrits ci-dessus démontrent que déléguer une partie des calculs d'un algorithme permet de faciliter son implémentation sur des périphériques de faible puissance. Cependant, cela nécessite une étude spécifique à chaque protocole afin de s'assurer que la sécurité n'est pas remise en cause.

Une façon de traiter plus rapidement les différents cas est de noter que les protocoles cryptographiques sont souvent conçus à partir des mêmes briques de base. Dans le cas des protocoles « anonymes », l'usage de preuves de connaissance est par exemple quasi systématique. Il peut donc être intéressant d'étudier une fois pour toutes la délégation de ces briques cryptographiques afin de construire plus rapidement les versions coopératives des protocoles les utilisant. C'est l'objet des travaux présentés dans l'article *Efficient Delegation of Zero-Knowledge Proofs of Knowledge in a Pairing-Friendly Setting* [CPS14], où nous nous sommes intéressés à la délégation de preuves de connaissance de logarithmes discrets dans les groupes bilinéaires.

Considérons, par exemple, le schéma de monnaie électronique divisible que nous venons de décrire dans la section 6.1.2. L'utilisateur doit y prouver la connaissance d'un scalaire x tel que $t_s = g_s^x$ et tel que la signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ vérifie $\sigma_4 = \sigma_2^x$. Cette situation, que l'on retrouve par exemple dans les constructions de [CPS10, BFG+13], correspond à ce que Kiayias, Tsiounis et Yung [KTY04] définirent comme un ensemble de relations de logarithmes discrets, que nous noterons DLRS (de l'anglais *Discret Logarithms Relations Set*).

6.2.1 Ensemble de Relations de Logarithmes Discrets

Définition 41. Un *ensemble de relations de logarithmes discrets* $\mathsf{R}(\alpha_1, \ldots, \alpha_m)$ sur un groupe $\mathbb{G}$ d'ordre p est un ensemble de r relations $R_1, \ldots, R_r$ définies par des objets $A_1, \ldots, A_w, V_1, \ldots, V_r \in \mathbb{G}$ et des variables $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_p$ tels que :

$$R_i : \ V_i = \prod_{j \in \mathcal{J}_i} A_{v_{i,j}}^{\alpha_j}$$

avec $\mathcal{J}_i \subseteq \{1, \ldots, m\}$ et $1 \leq v_{i,j} \leq w$ pour tout $1 \leq i \leq r$ et $j \in \mathcal{J}_i$.

Exemple 4: Si nous reprenons l'exemple du schéma de monnaie électronique, le DLRS $\mathsf{R}(x)$ contient deux relations R_1 et R_2 telles que :
- $R_1 : \ V_1 = A_1^x$
- $R_2 : \ V_2 = A_2^x$

avec $V_1 = t_s$, $V_2 = \sigma_4$, $A_1 = g_s$ et $A_2 = \sigma_2$.

Pour prouver connaissance des variables $\alpha_1, \ldots, \alpha_m$ impliquées dans un DLRS, on utilise généralement le Σ-protocole que nous décrivons dans la figure 6.3. Celui-ci est souvent appelé *protocole de Schnorr étendu* en référence au protocole de Schnorr (figure 3.2) dont il est issu.

Remarque 17. Les résultats que nous présentons dans cette section s'appliquent aux relations dans les groupes $\mathbb{G}_1$ et $\mathbb{G}_2$. Afin de simplifier la description du protocole nous ne décrivons cependant que le cas $\mathbb{G} = \mathbb{G}_1$.

Remarque 18. Le vérificateur des preuves de connaissance décrites dans les figures 6.3, 6.4 et 6.5 choisit le challenge c comme un élément aléatoire de $\mathbb{Z}_p$. Comme nous l'avons expliqué dans

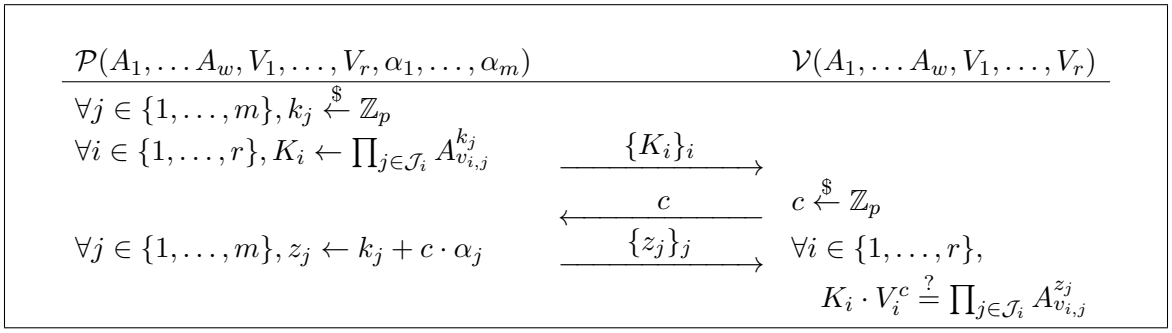


Figure 6.3 – Protocole de Schnorr étendu pour un DLRS $\mathsf{R}(x)$.

la section 3.3.2, le protocole ne vérifie alors la propriété de divulgation nulle que face à un vérificateur honnête. Cela est néanmoins suffisant lorsque la preuve de connaissance est rendue non-interactive à l'aide de la méthode Fiat-Shamir, ce qui est en général le cas des schémas décrits dans ce mémoire.

Il est possible d'éviter cette restriction en choisissant un challenge c dans un ensemble $\{0,1\}^\ell$ contenant un nombre polynomial d'éléments. Le protocole doit alors répété k fois (de sorte que $2^{k \cdot \ell}$ soit super-polynomial) pour en garantir la validité.

6.2.2 Un Premier Protocole

On peut noter que le coût, pour le prouveur $\mathcal{P}$, du protocole de Schnorr étendu augmente avec le nombre r de relations à prouver. Dans le cas de notre exemple, la carte SIM devrait, lorsqu'une valeur 2^ℓ est dépensée, effectuer 2 exponentiations pour prouver $\mathsf{R}(x)$. A priori, cela semble raisonnable. Cependant, en pratique, il est plutôt rare que le montant à payer soit une puissance de 2. Supposons par exemple qu'un utilisateur doive dépenser 10.23 €. La décomposition binaire de 1023 étant $\sum_{i=0}^{9} 2^i$, l'utilisateur va devoir effectuer des calculs pour 10 sommets (1 pour chaque puissance de 2 intervenant dans cette décomposition). Il doit donc calculer 10 éléments $t_{s_i} = g_{s_i}^x$ (où s_i correspond à la valeur 2^i) et prouver qu'ils sont bien formés. Le DLRS $\mathsf{R}(x)$ contient alors 11 relations (car il faut rajouter $\sigma_4 = \sigma_2^x$) et impose donc à la SIM de calculer 11 exponentiations si le protocole de la figure 6.3 est utilisé.

Afin de diminuer ce coût pour la carte SIM, nous avons proposé dans [CPS14] un moyen de déléguer cette preuve au téléphone. Notre protocole, qui s'applique à n'importe quel DLRS sur $\mathbb{G}_1$ ou sur $\mathbb{G}_2$, ne nécessite plus qu'une exponentiation par secret (indépendamment du nombre de relations) de la part de la carte SIM. Dans le cas de la dépense de 10.23 €, celle-ci n'a par exemple besoin d'effectuer qu'une seule exponentiation dans $\mathbb{G}_2$ afin de prouver $\mathsf{R}(x)$.

L'idée de notre construction est de confier au téléphone la connaissance de quelques éléments supplémentaires qui vont lui permettre de calculer la majeure partie de la preuve. Le point essentiel est que nous pouvons prouver que celui-ci n'apprendra aucune autre information lors de l'exécution du protocole. Par ailleurs, ces éléments ne lui permettent pas de prouver la connaissance des secrets sans l'aide de la SIM et nous montrerons, à l'aide des exemples de la section 6.1.2, qu'ils ne remettent pas en cause la sécurité de la version coopérative de bon nombre de schémas cryptographiques.

Intuition. Si nous reprenons le formalisme des DLRS, les relations intervenant dans notre dépense sont de la forme $V_i = A_i^\alpha$ pour $i \in \{1, \dots, r\}$. L'utilisation du protocole de Schnorr étendu pour prouver ces relations induit le calcul de r exponentiations. Plus précisément, lors du calcul des mises en gages, le prouveur va choisir un scalaire $k \xleftarrow{\$} \mathbb{Z}_p$ et calculer $K_i \leftarrow A_i^k$ pour tout $1 \leq i \leq r$.

Supposons maintenant que ces éléments soient désormais calculés dans $\mathbb{G}_T$, c'est-à-dire que,

$\forall i$, $K_i \leftarrow e(A_i^k, \widetilde{g})$ pour un générateur $\widetilde{g}$ de $\mathbb{G}_2$. La vérification de la preuve resterait possible en testant, $\forall i$, la validité des relations dans $\mathbb{G}_T$:

$$K_i \cdot e(V_i, \widetilde{g})^c \stackrel{?}{=} e(A_i, \widetilde{g})^z$$

Les éléments K_i peuvent alors également s'écrire $e(A_i, \widetilde{g}^k)$ en raison de la bilinéarité du couplage. La connaissance de $\widetilde{g}^k$ suffit donc à les calculer. Malheureusement, cet élément ne peut pas être envoyé au vérificateur sans perdre la propriété de divulgation nulle car celui-ci pourrait alors retrouver $\widetilde{g}^\alpha \leftarrow (g^z \cdot \widetilde{g}^{-k})^{c^{-1}}$. Si $\widetilde{g}^\alpha$ constitue en soi une information supplémentaire sur le secret α, celle-ci reste « mineure » (nous préciserons ce terme dans la remarque 19). En particulier, elle ne permet pas de prouver connaissance de α et donc de se substituer au prouveur. Nous confierons donc $\widetilde{g}^\alpha$ au téléphone, en supposant, comme dans la section 6.1.2, que celui-ci dispose d'un niveau de confiance supérieur à celui de n'importe quel vérificateur. Il va alors pouvoir rendre la preuve à divulgation nulle vis-à-vis de ce dernier, en calculant, pour tout $i \in \{1, \ldots, r\}$ les éléments $Z_i \leftarrow A_i^{b_i^{-1}}$ et $\widetilde{B}_i \leftarrow (\widetilde{g}^k)^{b_i}$ pour un scalaire b_i aléatoire. De son côté, le vérificateur peut reconstituer les mises en gage K_i en calculant, $\forall i$, $e(Z_i, \widetilde{B}_i) = e(A_i, \widetilde{g}^k) = K_i$.

Cependant, cette idée ne s'applique pas directement aux relations plus complexes, à savoir celles faisant intervenir plusieurs bases $A_{i,j}$ dans une même relation. En effet, considérons la relation :

$$V_i = \prod_{j \in \mathcal{J}} A_{v_{i,j}}^{\alpha_j}$$

L'envoi des éléments $Z_{i,j} \leftarrow A_{v_{i,j}}^{b_{i,j}^{-1}}$ et $\widetilde{B}_{i,j} \leftarrow (\widetilde{g}^{k_j})^{b_{i,j}}$ au vérificateur lui révèlerait les valeurs $e(A_{v_{i,j}}, \widetilde{g})^{\alpha_j}$. En effet :

$$(e(A_{v_{i,j}}, \widetilde{g})^{z_j} \cdot e(Z_{i,j}^{-1}, \widetilde{B}_{i,j}))^{c^{-1}} = e(A_{v_{i,j}}, \widetilde{g})^{\alpha_j}.$$

Pour éviter ce problème, nous allons « lier » entre eux tous les éléments $\widetilde{B}_{i,j}$, pour un même indice i. Nous obtenons alors le protocole décrit dans la figure 6.4, où *p.p.* désigne l'ensemble des objets $A_1, \ldots, A_w, V_1, \ldots, V_r$ intervenant dans le DLRS et où $\widetilde{g}$ est un générateur de $\mathbb{G}_2$.

SIM($p.p., \alpha_1, \ldots, \alpha_m$)		Téléphone($p.p.$)		$\mathcal{V}(p.p.)$
$\forall 1 \le j \le m$,		$\forall 1 \le i \le r$,		
$k_j \stackrel{\$}{\leftarrow} \mathbb{Z}_p, \widetilde{Z}_j \leftarrow \widetilde{g}^{k_j}$		$(b_{i,j}, t_{i,j}) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{2\cdot\lvert\mathcal{J}_i\rvert}$		
		$H_i \leftarrow \prod_{j \in \mathcal{J}_i} A_{v_{i,j}}^{t_{i,j}}$		
	$\xrightarrow{\{\widetilde{Z}_j\}_j}$	$\forall 1 \le i \le r, \forall j \in \mathcal{J}_i$,		
		$Z_{i,j} \leftarrow A_{v_{i,j}}^{b_{i,j}^{-1}}$		
		$\widetilde{B}_{i,j} \leftarrow (\widetilde{Z}_j \cdot \widetilde{g}^{t_{i,j}})^{b_{i,j}}$		
			$\xrightarrow{\{H_i\}_i, \{Z_{i,j}, \widetilde{B}_{i,j}\}_{i,j}}$	
	$\xleftarrow{c}$		$\xleftarrow{c}$	$c \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
$\forall 1 \le j \le m$,				
$z_j \leftarrow k_j + c \cdot \alpha_j$	$\xrightarrow{\{z_j\}_j}$		$\xrightarrow{\{z_j\}_j}$	$\forall 1 \le i \le r$,
			$e(H_i \cdot V_i^{-c} \prod_{j \in \mathcal{J}_i} A_{v_{i,j}}^{z_j}, \widetilde{g}) \stackrel{?}{=} \prod_{j \in \mathcal{J}_i} e(Z_{i,j}, \widetilde{B}_{i,j})$	

Figure 6.4 – Délégation de Preuves de Connaissance pour un DLRS $\mathsf{R}(\alpha_1, \ldots, \alpha_m)$.

La figure 6.4 décrit en réalité deux preuves de connaissance des secrets $\alpha_1, ..., \alpha_m$. La première se joue entre un prouveur constitué de la carte SIM et du téléphone et un vérificateur $\mathcal{V}$. La deuxième se joue entre la carte SIM et un vérificateur constitué du téléphone et de $\mathcal{V}$. Dans ce deuxième cas, le vérificateur, qui connait déjà les éléments contenus dans *p.p.*, n'apprend rien d'autre que l'ensemble $\{\widetilde{g}^{\alpha_j}\}_{j=1}^m$. Ces résultats sont énoncés par les deux théorèmes suivants (nous rappelons que la chaîne $y \in \{0,1\}^*$ a été définie dans la section 3.3.1 comme l'entrée du vérificateur d'un système de preuves).

Théorème 42. *Le protocole décrit dans la figure 6.4 est une preuve de connaissance des témoins $\alpha_1, \ldots, \alpha_m$ entre un prouveur (SIM + téléphone) et un vérificateur honnête ($\mathcal{V}$) prenant en entrée $y \leftarrow$ p.p..*

Théorème 43. *Le protocole décrit dans la figure 6.4 est une preuve de connaissance des témoins $\alpha_1, \ldots, \alpha_m$ entre un prouveur (SIM) et un vérificateur honnête (téléphone + $\mathcal{V}$) prenant en entrée $y \leftarrow p.p. \cup \{\widetilde{g}^{\alpha_j}\}_{j=1}^m$.*

Remarque 19. Notre protocole repose sur l'hypothèse qu'il est possible de confier les éléments $\widetilde{g}^{\alpha_j}$ au téléphone. Il est donc nécessaire de s'assurer des conséquences sur la sécurité en cas de corruption de ce périphérique. Là encore, cela va dépendre des cas considérés. Cependant, le théorème 43 permet d'accélérer significativement cette étude. Il garantit en effet qu'il suffit de s'assurer que la réduction construite dans les différentes preuves de sécurité soit capable de fournir les éléments $\widetilde{g}^{\alpha_j}$.

Dans le cas du protocole de dépense, nous avons $m = 1$ et $\alpha_1 = x$. La réduction du jeu de la traçabilité et celle du jeu de la non-diffamation doivent donc être capables de fournir $\widetilde{g}^x$. Dans le premier cas, cela ne pose aucun problème car la réduction connait le secret x. Dans le deuxième, cela nécessite simplement de remplacer l'hypothèse DL sur laquelle repose la propriété de non-diffamation par sa variante symétrique SDL.

De même, il est possible de prouver que le schéma d'attestation anonyme de [BFG+13] ou celui de signature de groupe de [BCN+10] restent sûrs lorsque notre protocole de délégation de preuves est utilisé.

Efficacité. Nous comparerons dans le tableau 6.1 à la fin de ce chapitre l'efficacité de notre protocole avec celui de la figure 6.3. Il est cependant important de noter que, bien que ce protocole augmente le coût pour le téléphone et pour le vérificateur, il diminue sensiblement le coût pour la carte SIM, le rendant indépendant du nombre de relations à prouver. Il se justifie donc pleinement dans le contexte actuel où l'écart de performances entre la carte SIM et les autres périphériques (tels que le téléphone) est considérable.

L'exemple de la dépense de $10,23 €$ illustre ce gain pour la SIM qui n'a plus qu'une exponentiation à effectuer dans $\mathbb{G}_2$ contre 11 dans $\mathbb{G}_1$ si le protocole de Schnorr étendu est utilisé. De plus, la phase de mise en gage (à savoir l'évaluation de $\widetilde{g}^k$ pour un scalaire k aléatoire) peut être pré-calculée car elle ne dépend que du paramètre $\widetilde{g}$ du système. Ce n'est pas aussi évident pour les mises en gage du protocole de Schnorr étendu car celles-ci dépendent des éléments g_{s_i}. Leur pré-calcul nécessite donc de deviner les sommets s_i qui seront dépensés lors de la prochaine dépense.

6.2.3 Preuves de Sécurité

Nous prouvons dans cette section les théorèmes 42 et 43 que nous venons d'énoncer. Les preuves de la complétude et de l'existence d'un extracteur seront communes à ces deux théorèmes. Seule la propriété de divulgation nulle nécessite d'étudier séparément chaque cas.

Complétude

Si $\forall j \in \{1,\dots,m\}$, $z_j \leftarrow k_j + c \cdot \alpha_j$, alors, $\forall i$:

$$\begin{aligned} e(H_i \cdot V_i^{-c} \prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{z_j}, \widetilde{g}) &= e(\prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{t_{i,j}} \cdot (\prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{\alpha_j})^{-c} \cdot \prod_{j\in\mathcal{J}_i} (A_{v_{i,j}}^{k_j} \cdot A_{v_{i,j}}^{c\cdot\alpha_j}), \widetilde{g}) \\ &= e(\prod_{j\in\mathcal{J}_i} (A_{v_{i,j}}^{t_{i,j}} \cdot A_{v_{i,j}}^{k_j}), \widetilde{g}) = e(\prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{k_j+t_{i,j}}, \widetilde{g}) \\ &= \prod_{j\in\mathcal{J}_i} e(A_{v_{i,j}}, \widetilde{g}^{k_j+t_{i,j}}) = \prod_{j\in\mathcal{J}_i} e(A_{v_{i,j}}^{b_{i,j}^{-1}}, (\widetilde{Z}_j \cdot \widetilde{g}^{t_{i,j}})^{b_{i,j}}) \\ &= \prod_{j\in\mathcal{J}_i} e(Z_{i,j}, \widetilde{B}_{i,j}) \end{aligned}$$

La propriété de complétude est donc satisfaite.

Existence d'un extracteur

Supposons qu'un prouveur soit capable, pour une même mise en gage $(\{H_i\}_i, \{Z_{i,j}, \widetilde{B}_{i,j}\}_{i,j})$, de répondre $\{z_j\}_j$ à un challenge c et $\{z'_j\}_j$ à un challenge c'. Nous avons alors, $\forall i$:

$$e(H_i \cdot V_i^{-c} \prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{z_j}, \widetilde{g}) = e(Z_{i,j}, \widetilde{B}_{i,j}) = e(H_i \cdot V_i^{-c'} \prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{z'_j}, \widetilde{g}).$$

Par conséquent,

$$e(V_i^{c-c'}, \widetilde{g}) = \prod_{j\in\mathcal{J}_i} e(A_{v_{i,j}}^{z_j-z'_j}, \widetilde{g})$$

et donc $e(V_i \cdot \prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{\frac{z_j-z'_j}{c'-c}}, \widetilde{g}) = 1_{\mathbb{G}_T}$. Le couplage étant non-dégénéré, on en déduit que :

$$V_i = \prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{\frac{z_j-z'_j}{c-c'}}.$$

Il est donc possible d'extraire les éléments $\alpha_j \leftarrow \frac{z_j-z'_j}{c-c'}$ tels que $V_i = \prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{\alpha_j}$.

Remarque 20. Dans le cas du théorème 43, la mise en gage est constituée de l'ensemble $\{\widetilde{Z}_j\}_j$. Cela correspond au cas particulier où $H_i = 1_{\mathbb{G}_1}$, $Z_{i,j} = A_{i,j}$ et $\widetilde{B}_{i,j} = \widetilde{Z}_j$ $\forall i \in \{1,\dots,r\}$ et $\forall j \in \{1,\dots,m\}$. L'extracteur que nous venons de décrire fonctionne alors de la même manière.

Divulgation nulle vis-à-vis du téléphone

Dans le cas du théorème 43, la chaîne y prise en entrée par le vérificateur (qui inclut le téléphone) contient *p.p.* ainsi que les éléments $\widetilde{T}_j \leftarrow \widetilde{g}^{\alpha_j}$ pour tout $j \in \{1,\dots,m\}$. Le simulateur, prenant également y en entrée, procède alors comme suit :

1. il génère, $\forall j$, $c \xleftarrow{\$} Z_p$ et $z_j \xleftarrow{\$} \mathbb{Z}_p$;
2. il calcule, $\forall j$, $\widetilde{Z}_j \leftarrow \widetilde{g}^{z_j} \cdot \widetilde{T}_j^{-c}$;
3. il retourne $(\{\widetilde{Z}_j\}_j, c, \{z_j\}_j)$.

Les éléments $(\{\widetilde{Z}_j\}_j, c, \{z_j\}_j)$ sont alors indistinguables de la trace d'une exécution du protocole entre un prouveur et un vérificateur honnête (défini dans la section 3.3.2).

Divulgation nulle vis-à-vis de $\mathcal{V}$

Dans le cas du théorème 42, la chaîne y prise en entrée par le vérificateur $\mathcal{V}$ contient uniquement *p.p.*. Le simulateur agit alors de la manière suivante :

1. il génère, $\forall j$, $c \xleftarrow{\$} Z_p$ et $z_j \xleftarrow{\$} \mathbb{Z}_p$;
2. il calcule, $\forall i$, $K_i \leftarrow \prod_{j\in\mathcal{J}_i}(A_{v_{i,j}}^{z_j} \cdot V_i^{-c})$;
3. il génère des scalaires $u_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ ainsi que des éléments $U_{v_{i,j}} \xleftarrow{\$} \mathbb{G}_1^*$ pour tout $i \in \{1,\dots,r\}$ et $j \in \mathcal{J}_i$;
4. il calcule, $\forall i$, $H_i \leftarrow K_i^{-1} \cdot \prod_{j\in\mathcal{J}_i} U_{v_{i,j}}$ et $(Z_{i,j}, \widetilde{B}_{i,j}) \leftarrow (U_{v_{i,j}}^{u_{i,j}^{-1}}, \widetilde{g}^{u_{i,j}})$;
5. il retourne $(\{H_i\}_i, \{Z_{i,j}, \widetilde{B}_{i,j}\}_{i,j}, c, \{z_j\}_j)$.

Ces éléments vérifient bien l'équation

$$e(H_i \cdot V_i^{-c} \prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{z_j}, \widetilde{g}) \stackrel{?}{=} \prod_{j\in\mathcal{J}_i} e(Z_{i,j}, \widetilde{B}_{i,j}).$$

Il reste à montrer qu'ils sont également indistinguables de la trace d'une exécution normale du protocole. Rappelons que cette dernière contient $H_i = \prod_{j\in\mathcal{J}_i} A_{v_{i,j}}^{t_{i,j}}$, $Z_{i,j} = A_{v_{i,j}}^{b_{i,j}^{-1}}$, $\widetilde{B}_{i,j} = (\widetilde{Z}_j \cdot \widetilde{g}^{t_{i,j}})^{b_{i,j}} = \widetilde{g}^{b_{i,j}(k_j+t_{i,j})}$, c et z_j.

Si nous posons $u'_{i,j} = b_{i,j}(k_j + t_{i,j})$ et $U'_{v_{i,j}} = A_{v_{i,j}}^{t_{i,j}+k_j}$, alors $\widetilde{B}_{i,j} = \widetilde{g}^{u'_{i,j}}$, $Z_{i,j} = (U'_{v_{i,j}})^{(u'_{i,j})^{-1}}$ et $V_i^c \cdot \prod_{j\in\mathcal{J}_i}(A_{v_{i,j}}^{-z_j} \cdot U'_{v_{i,j}}) = \prod_{j\in\mathcal{J}_i}(A_{v_{i,j}}^{-k_j} \cdot A_{v_{i,j}}^{t_{i,j}+k_j}) = H_i$.

Comme $t_{i,j}$ est choisi aléatoirement, les éléments $U'_{i,j}$ et $U_{i,j}$ sont distribués de la même manière. De même pour $u'_{i,j}$ et $u_{i,j}$ car $b_{i,j}$ est aléatoire. La simulation est donc valide.

6.2.4 Un Nouveau Protocole

Le protocole de la figure 6.4, publié dans [CPS14], peut être amélioré afin de réduire sensiblement sa complexité pour le téléphone. L'idée est de décomposer le scalaire k_j intervenant dans la mise en gage en deux parties, c'est-à-dire que $k_j = u_j - t_j$ pour deux scalaires aléatoires u_j et t_j. La carte SIM va désormais calculer les éléments $\widetilde{Z}_i \leftarrow \widetilde{g}^{u_j}$ qui pourront directement être envoyés à $\mathcal{V}$ car ils n'apportent, à eux seuls, aucune information sur k_j et donc sur α_j. Les scalaires t_j seront quant à eux envoyés au téléphone qui ne pourra pas en tirer plus d'informations sur les secrets que dans le protocole précédent. L'ensemble de ce nouveau protocole est décrit dans la figure 6.5 où *p.p.* est défini de la même manière que précédemment.

Ce nouveau protocole offre de bien meilleures performances que le précédent tout en vérifiant les mêmes propriétés de sécurité. Celles-ci sont précisées par les deux théorèmes ci-dessous.

Théorème 44. *Le protocole décrit dans la figure 6.5 est une preuve de connaissance des témoins $\alpha_1,\dots,\alpha_m$ entre un prouveur (SIM + téléphone) et un vérificateur honnête ($\mathcal{V}$) prenant en entrée $y \leftarrow p.p.$.*

Théorème 45. *Le protocole décrit dans la figure 6.5 est une preuve de connaissance des témoins $\alpha_1,\dots,\alpha_m$ entre un prouveur (SIM) et un vérificateur honnête (téléphone + $\mathcal{V}$) prenant en entrée $y \leftarrow p.p. \cup \{\widetilde{g}^{\alpha_j}\}_{j=1}^m$.*

La preuve de l'existence d'un extracteur est similaire à celle de la section 6.2.3. La complétude et les propriétés de divulgation nulle sont prouvées ci-dessous.

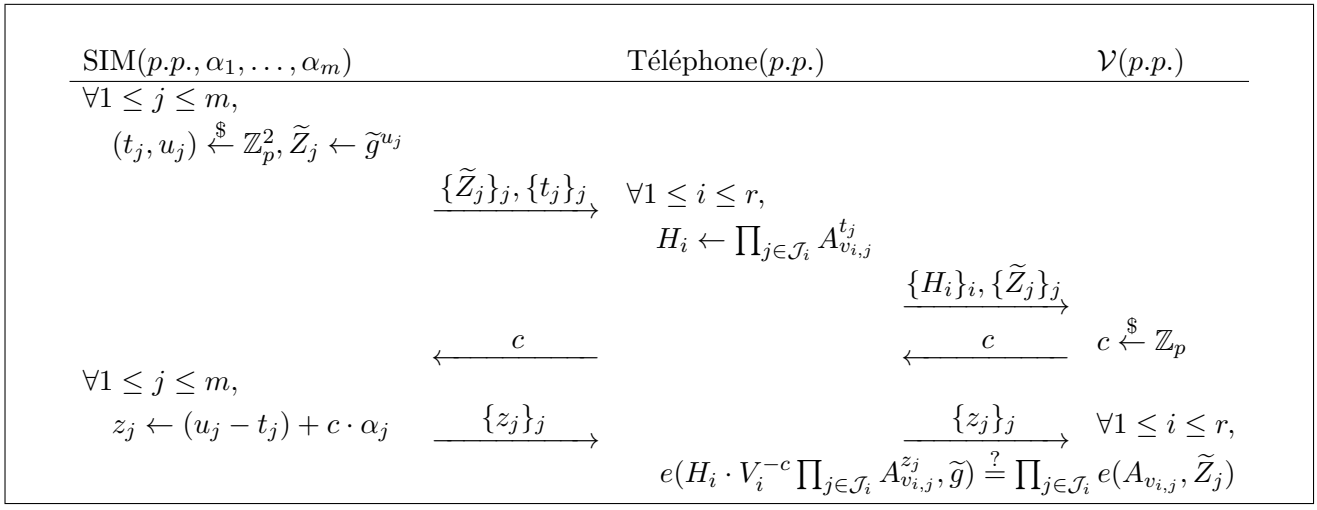


Figure 6.5 – Délégation de Preuves de Connaissance pour un DLRS $\mathsf{R}(\alpha_1, \ldots, \alpha_m)$ nécessitant moins de calculs.

Complétude. Pour tout $i \in \{1, \ldots r\}$, nous avons :

$$\begin{aligned} e(H_i \cdot V_i^{-c} \prod_{j \in \mathcal{J}_i} A_{v_{i,j}}^{z_j}, \widetilde{g}) &= e(V_i^{-c} \prod_{j \in \mathcal{J}_i} A_{v_{i,j}}^{z_j + t_j}, \widetilde{g}) \\ &= e(V_i^{-c} \prod_{j \in \mathcal{J}_i} A_{v_{i,j}}^{u_j + c \cdot \alpha_j}, \widetilde{g}) \\ &= e(\prod_{j \in \mathcal{J}_i} A_{v_{i,j}}^{u_j}, \widetilde{g}) \\ &= \prod_{j \in \mathcal{J}_i} e(A_{v_{i,j}}, \widetilde{Z}_j) \end{aligned}$$

Divulgation nulle vis-à-vis du téléphone. Dans le cas du théorème 45, le téléphone connait les éléments $\widetilde{T}_j \leftarrow \widetilde{g}^{\alpha_j}$ pour tout $j \in \{1, \ldots, m\}$. Le simulateur fonctionne alors de la manière suivante :

1. il génère, $\forall j$, $c \xleftarrow{\$} Z_p$ et $(z_j, t_j) \xleftarrow{\$} \mathbb{Z}_p^2$;
2. il calcule, $\forall j$, $\widetilde{Z}_j \leftarrow \widetilde{g}^{z_j + t_j} \cdot T_j^{-c}$;
3. il retourne $(\{\widetilde{Z}_j\}_j, \{t_j\}_j, c, \{z_j\}_j)$.

Ces éléments sont alors indistinguables d'une trace de l'exécution du protocole. En effet, les scalaires u_j étant choisis aléatoirement dans le protocole, les valeurs z_j retournées par le simulateur et celles retournées par la carte SIM présentent la même distribution.

Divulgation nulle vis-à-vis de $\mathcal{V}$. Le simulateur, qui n'a ici accès qu'à $p.p.$, exécute les étapes suivantes.

1. il génère, $\forall j$, $c \xleftarrow{\$} Z_p$ et $z_j \xleftarrow{\$} \mathbb{Z}_p$;
2. il génère $u_j \xleftarrow{\$} \mathbb{Z}_p$ et calcule, $\forall j$, $\widetilde{Z}_j \leftarrow \widetilde{g}^{u_j}$;
3. il calcule, $\forall i$, $H_i \leftarrow V_i^c \prod_{j \in \mathcal{J}_i} A_{v_{i,j}}^{u_j - z_j}$;
4. il retourne $(\{\widetilde{Z}_j\}_j, \{H_i\}_i, c, \{z_j\}_j)$.

Dans ce protocole, les éléments H_i sont construits comme $H_i \leftarrow \prod_{j \in \mathcal{J}_i} A_{v_{i,j}}^{t_j}$ avec $t_j = (u_j - z_j) + c \cdot \alpha_j$. Puisque le simulateur a généré aléatoirement les scalaires z_j, les éléments H_i qu'il a produits sont distribués de manière identique à ceux du protocole. Par conséquent, les éléments qu'il a retournés à l'étape 4 sont indistinguables d'une trace d'une exécution du protocole.

6.2.5 Efficacité

	Prouveur		Vérificateur
	SIM	Téléphone	
Schnorr étendu (Fig 6.3)	$n\ \mathtt{E}_{\mathbb{G}}$	-	$(n+r)\ \mathtt{E}_{\mathbb{G}}$
Nous (Fig 6.4)	$m\ \mathtt{E}_{\mathbb{G}_2}$	$2n\ \mathtt{E}_{\mathbb{G}_1} + 2n\ \mathtt{E}_{\mathbb{G}_2}$	$(n+r)\ \mathtt{E}_{\mathbb{G}_1} + (n+r)\ \mathtt{P}$
Nous (Fig 6.5)	$m\ \mathtt{E}_{\mathbb{G}_2}$	$n\ \mathtt{E}_{\mathbb{G}_1}$	$(n+r)\ \mathtt{E}_{\mathbb{G}_1} + (n+r)\ \mathtt{P}$

Table 6.1 – Complexité des différents protocoles de preuves de DLRS pour m variables et r relations R_i impliquant chacune $|\mathcal{J}_i|$ éléments. L'entier $n = \sum_{i=1}^{r} |\mathcal{J}_i|$ est en particulier supérieur (souvent de beaucoup) à m. La notation $\mathtt{E}_{\mathbb{G}}$ désigne une exponentiation dans le groupe $\mathbb{G}$, $\mathtt{E}_{\mathbb{G}_1}$ (resp. $\mathtt{E}_{\mathbb{G}_2}$) une exponentiation dans le groupe $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) et $\mathtt{P}$ un calcul de couplages.

Le tableau 6.1 compare, pour chaque entité, la complexité du protocole de Schnorr étendu à celles des deux protocoles que nous avons introduits dans ce chapitre. Ces derniers permettent tous deux de diminuer sensiblement le coût pour la carte SIM, le rendant indépendant du nombre de relations à prouver. Cependant, cela se traduit, pour le premier protocole (figure 6.4), par une augmentation significative du nombre d'opérations à effectuer par le téléphone. Ce défaut est corrigé par notre deuxième protocole qui doit donc être privilégié pour déléguer une preuve de DLRS.

Chapitre 7

Délégation d'Opérations Mathématiques

Sommaire

Nous nous intéressons dans ce chapitre à la délégation d'opérations mathématiques au travers des exemples de l'exponentiation dans un groupe fini et du couplage bilinéaire. Nous présentons les principaux protocoles de l'état de l'art ainsi que les résultats introduits dans l'article *Delegating a Pairing can be Both Secure and Efficient* [CDS14], publié lors de la conférence ACNS 2014, que nous avons cosignés avec Sébastien Canard et Julien Devigne.

7.1 Introduction

Nous avons étudié, dans le chapitre précédent, la délégation de calculs cryptographiques à l'échelle d'un algorithme. Celle-ci consiste à découper ce dernier en une série d'opérations qui vont être réparties entre différentes entités en fonction du niveau de confiance dont elles bénéficient. Dans l'idéal, l'algorithme en question est une brique cryptographique fréquemment utilisée par des constructions plus complexes. Sa version coopérative peut alors servir pour chacune de ces dernières. C'est le cas, par exemple, des preuves de connaissance de secrets impliqués dans un DLRS que nous avons traité dans la section 6.2.

Il est possible d'aller plus loin dans ce raisonnement en s'intéressant directement aux briques élémentaires d'un algorithme, à savoir les opérations mathématiques qui le composent. Une version coopérative, efficace et sûre d'une d'entre elles permettrait en effet d'améliorer le temps d'exécution d'une multitude de schémas cryptographiques. Cela explique le très grand nombre de travaux (dont nous citons certains exemples ci-dessous) menés sur ce sujet.

La première préoccupation de la délégation de calculs est de diminuer la charge de l'entité devant effectuer l'opération. Elle se justifie donc seulement pour des opérations complexes. Déléguer une opération qu'un périphérique peut effectuer en 1 ms n'a par exemple pas vraiment de sens car

le temps nécessaire à la transmission des données au délégataire risque, à lui seul, d'être supérieur. La communauté cryptographique s'est donc, dans un premier temps, essentiellement concentrée sur la délégation de l'exponentiation (*e.g.* [MKI90, BQ95, HL05, CLM+12, WWW+14]). Celle-ci a en effet longtemps été l'opération la plus coûteuse de la cryptographie sur des groupes d'ordre fini. L'introduction en cryptographie du couplage bilinéaire [Jou00], dont la complexité est encore plus grande, a quelque peu changé la donne et entraîné de nombreux travaux sur la délégation de cette « nouvelle » opération (*e.g.* [GL05, CCM+10, CDS14, GV14]).

Cependant, la délégation ne doit pas se faire au prix de la sécurité. Comme dans le chapitre précédent, les propriétés de sécurité à satisfaire dépendent essentiellement des cas d'usage mais elles sont généralement rattachées à deux problèmes majeurs : la confidentialité et la vérifiabilité.

Le problème de la confidentialité consiste à s'assurer que le délégataire n'apprenne rien (ou juste une quantité limitée d'informations) sur certaines (voire toutes) des entrées de l'opération. Dans le cas de l'exponentiation, il se rencontre par exemple lors de la génération d'une signature RSA [RSA78] où l'exposant constitue le secret du signataire. Dans le cas du couplage, on peut le retrouver lors du déchiffrement du schéma de Boneh et Franklin [BF01] où l'une des entrées est la clé secrète de l'utilisateur.

Le problème de la vérifiabilité consiste, lui, à s'assurer que le calcul effectué par le délégataire est correct. Dans certains cas (par exemple, ceux que nous avons rencontrés dans le chapitre précédent), une valeur erronée n'aura pas d'autres conséquences que de faire échouer l'exécution du protocole. Malheureusement, cela n'est plus vrai si l'opération vise à tester la validité d'un certain élément public, comme c'est le cas lors de la vérification d'une signature. En effet, une mauvaise valeur pourrait conduire l'entité déléguant l'opération à accepter une signature incorrecte comme valide, et provoquer ainsi de graves problèmes de sécurité.

Concevoir un protocole efficace de délégation d'opérations sans faire aucune concession sur la sécurité est loin d'être évident. Le nombre de travaux menés sur le sujet l'illustre d'ailleurs parfaitement. En ce qui concerne l'exponentiation et le couplage, il n'existe, pour l'instant, aucun schéma qui soit pleinement satisfaisant sur le plan de l'efficacité et de la sécurité. Cependant, les solutions existantes offrent différents compromis qui peuvent être intéressants en fonction des cas d'usages. Nous présentons dans ce chapitre les principales constructions de l'état de l'art mais insistons sur le fait que comparer directement leur efficacité n'a pas toujours de sens car elles vérifient rarement les mêmes propriétés de sécurité.

7.2 Délégation d'Exponentiations

L'exponentiation est l'une des opérations fondamentales de la cryptographie asymétrique. On la retrouve par exemple dans l'échange de clés Diffie-Hellman [DH76], le cryptosystème RSA [RSA78] et le chiffrement El Gamal [ElG84]. Si son exécution sur un ordinateur, ou même sur un téléphone mobile, n'est pas un problème (*e.g.* [BCN14, SR13]), il n'en va pas de même pour des périphériques moins puissants, tels que des cartes à puces. Cet écart de performances, toujours d'actualité, a favorisé la production d'une multitude de travaux sur les moyens de déléguer cette opération.

La construction de protocoles de délégation d'exponentiations a connu dans un premier temps plusieurs échecs, par exemple les solutions de [MKI90, BQ95] qui furent cassées dans [PW93, NS98, NS01]. Intuitivement, la difficulté de concevoir de tels protocoles provient du fait que le cryptographe se retrouve privé de l'un ses principaux outils, à savoir l'exponentiation. Il doit alors utiliser d'autres techniques, qui se sont souvent révélées vulnérables.

Afin de rester génériques, les protocoles de délégation d'exponentiations évitent le plus souvent de faire des hypothèses sur le délégataire. Ils considèrent donc un unique délégataire contre lequel ils devront assurer la confidentialité ou la vérifiabilité, voire les deux. Ce modèle, qui est le plus universel, ne permet pour l'instant que des constructions à l'efficacité modérée [vDCG+06,

WWW+14]. Hohenberger et Lysyanskaya [HL05] proposèrent un nouveau modèle où l'entité déléguant les calculs a cette fois-ci accès à deux délégataires qui ne peuvent pas communiquer entre eux. Ils l'utilisèrent pour construire un protocole efficace mais sous l'hypothèse supplémentaire qu'au moins un des deux délégataires est honnête.

Nous présentons dans cette section ces deux approches et discutons leurs limites. Nous appelons la première le « modèle du délégataire unique » par opposition au « modèle des deux délégataires » introduit par Hohenberger et Lysyanskaya.

7.2.1 Modèle du Délégataire Unique

Comme nous l'avons expliqué, les premiers schémas dans le modèle du délégataire unique ont été cassés par la suite. Il existe aujourd'hui deux constructions qui sont encore considérées comme sûres dans ce modèle.

La première est due à Van Dijk *et al.* [vDCG+06] qui proposèrent deux protocoles assurant la vérifiabilité de la délégation, mais pas sa confidentialité. Malheureusement, leur complexité reste trop proche de celle d'une exponentiation classique. Leur protocole pour une base variable nécessite par exemple d'effectuer deux exponentiations avec des exposants de taille inférieure à λ, le paramètre de sécurité souhaité. Pour le groupe de points d'une courbe elliptique, dont l'ordre est en général de taille 2λ, le coût est équivalent.

Récemment, Wang *et al.* [WWW+14] proposèrent une nouvelle construction assurant à la fois la confidentialité et la vérifiabilité. Malheureusement, leur protocole pour déléguer une exponentiation, que nous décrivons dans la figure 7.1, nécessite une exponentiation avec un exposant de taille supérieure à λ, ce qui là encore limite la portée de leur résultat. De plus, la vérifiabilité n'est assurée qu'avec une probabilité de $\frac{1}{2}$, ce qui n'est pas acceptable dans bon nombre de cas (par exemple si ce protocole sert à vérifier une signature). Pour atteindre un niveau de sécurité satisfaisant, il est donc nécessaire de répéter ce protocole un grand nombre de fois, ce qui nuit considérablement à son efficacité.

Remarque 21. Comme l'ensemble des schémas décrits dans ce chapitre, ce protocole est divisé en deux phases. La première regroupe l'ensemble des calculs ne dépendant pas de la base ou de l'exposant impliqués dans l'opération. Ils peuvent donc être effectués à l'avance (d'où le nom de précalculs) et ne sont pas pris en compte dans l'estimation de la complexité. La deuxième phase regroupe le reste des calculs, c'est-à-dire ceux pour lesquels la connaissance de U ou de a est indispensable.

On peut noter que la phase 1 nécessite de générer un grand nombre d'éléments de la forme (k_i, g^{k_i}) pour un scalaire k_i aléatoire. Ce problème a notamment été étudié par Schnorr [Sch90, Sch91] mais les solutions proposées ont par la suite été cassées [dR91, dR97]. Il est néanmoins possible de générer une telle paire plus efficacement que par une exponentiation, comme le prouvèrent Boyko, Peinado et Venkatesan [BPV98] dont le travail fut par la suite amélioré dans [NSS00] et [WWW+14].

7.2.2 Modèle des Deux Délégataires

Partant du constat que les protocoles existants souffraient soit de problèmes d'efficacité, soit de problèmes de sécurité, Hohenberger et Lysyanskaya [HL05] proposèrent une approche totalement différente en supposant l'accès à deux délégataires, dont l'un est honnête, ne pouvant pas communiquer entre eux.

Cette hypothèse permet d'atteindre une bien meilleure efficacité comme l'illustre la construction de la figure 7.2. Intuitivement, la confidentialité repose sur le fait que la base et l'exposant intervenant dans l'exponentiation sont scindés en deux parties qui seront chacune envoyée à l'un des délégataires. L'entité déléguant l'opération n'aura alors qu'à « rassembler » les éléments qui

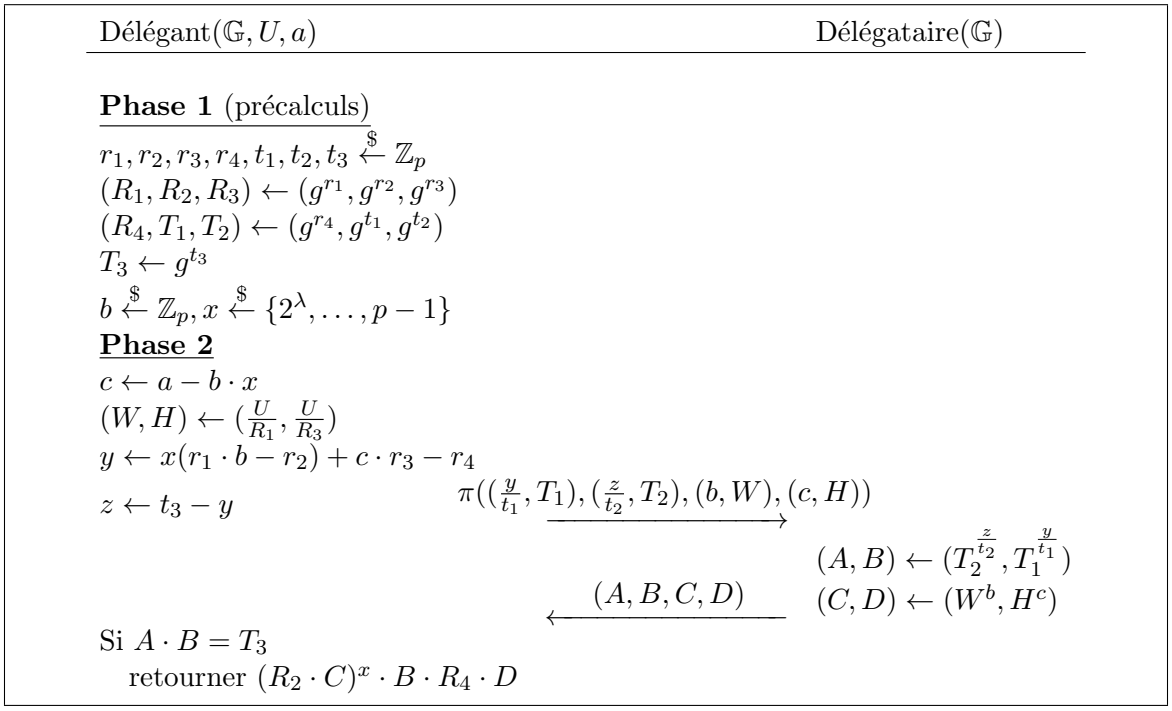


Figure 7.1 – Protocole de délégation d'exponentiations de Wang *et al.* Le calcul délégué est celui de U^a, où U un élément du groupe $\mathbb{G}$ d'ordre p et a un scalaire secret. La fonction π désigne une permutation aléatoire (voir remarque 22).

lui seront retournés pour reconstituer le résultat. Les deux délégataires seront quant à eux incapable d'agir de même car ils sont supposés ne pas pouvoir communiquer entre eux. La vérifiabilité s'obtient en leur demandant d'effectuer en plus des opérations (les mêmes pour les deux) qui n'ont pas d'autres buts que de tester leur honnêteté. Là encore, l'absence de communication rend difficile une collusion visant à retourner les mêmes valeurs erronées.

Cependant, il convient de préciser que les hypothèses faites par ce modèle sont particulièrement fortes et ne sont pas toujours vérifiées en pratique. L'accès à deux délégataires ne pouvant communiquer entre eux semble en effet impossible pour une carte SIM, le *trusted platform module* (TPM) embarqué dans un ordinateur ou une étiquette RFID.

Du point de vue de l'efficacité, ce protocole offre des performances incomparables avec celles des solutions utilisant un modèle de sécurité classique. Le délégant n'a en effet besoin d'effectuer que 5 multiplications (en excluant les précalculs) dans le groupe $\mathbb{G}$ pour déléguer une exponentiation. Il est cependant important de noter que la vérifiabilité n'est garantie qu'avec une probabilité de $\frac{1}{2}$, ce qui impose donc de répéter plusieurs fois le protocole pour atteindre un niveau de sécurité satisfaisant. Chen et al [CLM+12] ont récemment proposé une optimisation pour porter cette probabilité à $\frac{2}{3}$.

Remarque 22. La vérifiabilité des protocoles décrits dans les figures 7.1 et 7.2 repose sur le fait que le délégataire n'est pas capable de distinguer, parmi les éléments qu'il reçoit, ceux qui serviront à reconstituer U^a de ceux qui serviront à tester la validité du calcul. Il est par conséquent indispensable que l'ordre dans lequel les éléments sont envoyés ne révèlent pas cette information. Le délégant doit donc appliquer une permutation aléatoire π sur chaque ensemble avant leur transmission.

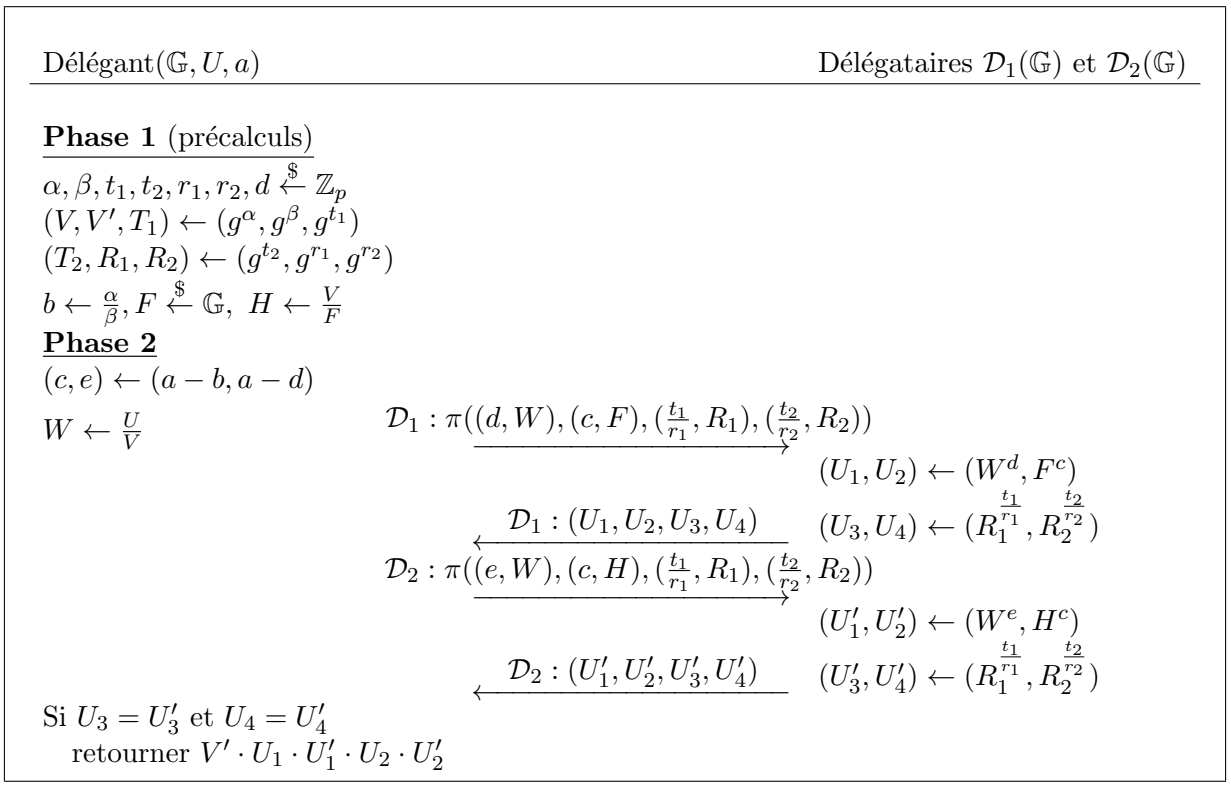


Figure 7.2 – Protocole de délégation d'exponentiations de Hohenberger et Lysyanskaya. L'entité de gauche délègue le calcul de U^a à deux délégataires $\mathcal{D}_1$ et $\mathcal{D}_2$. La fonction π désigne une permutation aléatoire (voir remarque 22).

7.3 Délégation de Couplages

L'ensemble des possibilités offertes par la cryptographie bilinéaire a un coût, celui de l'évaluation d'un (voire plusieurs) couplage. Cette complexité dépend bien évidemment de la courbe utilisée ainsi que de l'algorithme choisi pour l'implémenter (voir section 2.2.2). Néanmoins, une étude récente [BCN14] montre qu'elle est au moins 7 fois supérieure à celle d'une exponentiation dans le groupe $\mathbb{G}_1$ pour un choix de paramètres standards. Ce rapport est même de l'ordre de 20 pour des niveaux de sécurité supérieurs, tels que ceux offerts par les courbes KSS-18 [KSS08] et BLS-24 [BLS03], qui devront être privilégiés dans le futur. L'étude de la délégation de couplages est donc justifiée par cette importante complexité, souvent hors de portée d'un périphérique peu puissant.

Les premiers travaux sur ce sujet ont été publiés par Girault et Lefranc [GL05]. Leur protocole, que nous rappelons dans la figure 7.3, garantit la confidentialité d'un couplage prenant en entrée 1 ou 2 points secrets. Ils furent suivis par Chevallier-Mames *et al.* [CCM+05, CCM+10] qui proposèrent plusieurs protocoles satisfaisant la vérifiabilité ainsi que, dans certains cas, la confidentialité. Malheureusement, ceux-ci souffrent d'une complexité élevée, ce qui peut remettre en cause l'intérêt de la délégation. Kang *et al.* [KLP05] atténuèrent légèrement cette complexité mais, comme nous l'avons fait remarquer dans [CDS14], au prix d'un affaiblissement de la vérifiabilité. Plus précisément, pour les solutions décrites dans [KLP05], celle-ci n'est assurée qu'à condition que les points pris en entrée par le couplage soient issus d'un ensemble suffisamment grand. En effet, si N est la taille de cet ensemble, la vérifiabilité du protocole peut être cassée avec une probabilité d'au moins $1/N$. Cela peut par exemple poser problème si l'une des entrées du couplage est un certificat appartenant à une liste publique de taille polynomiale.

Dans l'article *Delegating a Pairing can be Both Secure and Efficient* [CDS14], nous avons montré que la vérifiabilité pouvait en fait être obtenue avec une complexité similaire à celle des solutions assurant la confidentialité. Avant de présenter notre construction, nous rappelons celles de [GL05] et [CCM+05] pour faciliter la comparaison.

Très récemment, deux nouveaux protocoles [GV14, TZR15] ont été publiés. Le premier assure la confidentialité du couplage (mais à condition que l'un des points soit public), offrant de meilleures performances que [GL05] mais en augmentant le volume de données à échanger avec le délégataire. Il s'agit donc d'un compromis qui peut être intéressant dans le cas où les transmissions peuvent s'effectuer en un temps négligeable. Le deuxième protocole applique le modèle des deux délégataires de Hohenberger et Lysyanskaya au couplage. Il présente donc les avantages et les défauts propres à ce modèle. Il est par conséquent incomparable aux autres schémas de l'état de l'art.

7.3.1 État de l'Art

Protocole de Girault et Lefranc

La figure 7.3 décrit le protocole de Girault et Lefranc qui fut le premier à assurer la confidentialité du couplage. Celle-ci est immédiate à prouver car le délégataire ne reçoit que des valeurs aléatoires de $\mathbb{G}_1$ et $\mathbb{G}_2$. Le coût pour les différentes entités, que nous récapitulons dans le tableau 7.1 en fin de chapitre, est assez faible, le délégant n'ayant qu'à effectuer une exponentiation dans chacun des groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$. On peut cependant remarquer que rien n'empêche le délégataire de retourner une mauvaise valeur de α. Le protocole n'est donc pas vérifiable.

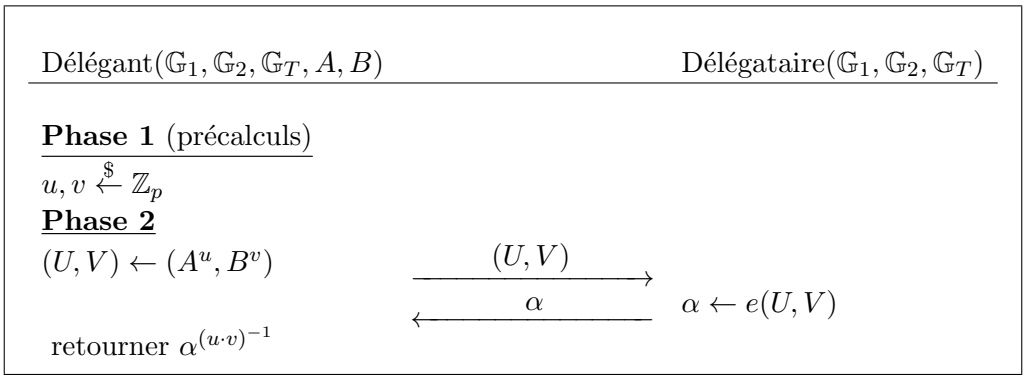


Figure 7.3 – Protocole de délégation du couplage $e(A, B)$ de Girault et Lefranc, où $A \in \mathbb{G}_1$ et $B \in \mathbb{G}_2$ sont des points secrets.

Protocole de Chevallier-Mames *et al.*

La figure 7.4 décrit le protocole de Chevallier-Mames *et al.* garantissant la vérifiabilité du couplage de deux points publics. Il suppose la connaissance, par le délégant, de la valeur $\rho \leftarrow e(G_1, G_2)$ pour deux points publics $G_1 \in \mathbb{G}_1$ et $G_2 \in \mathbb{G}_2$. La façon dont cette entité obtient la connaissance de ce couplage est indépendante du protocole. Celui-ci peut avoir été calculé une fois pour toutes par le délégant, ou alors avoir été chargé dans sa mémoire par une entité de confiance. Dans ce dernier cas, il n'est plus nécessaire d'implémenter l'algorithme de couplage sur ce périphérique ce qui, en plus des gains de performances, peut justifier l'usage d'un protocole de délégation.

Tester la validité des calculs retournés par le délégataire impose l'évaluation de plusieurs exponentiations dans $\mathbb{G}_T$ ainsi que de vérifier l'appartenance de α_1, α_2 et α_3 au groupe $\mathbb{G}_T$. Nous

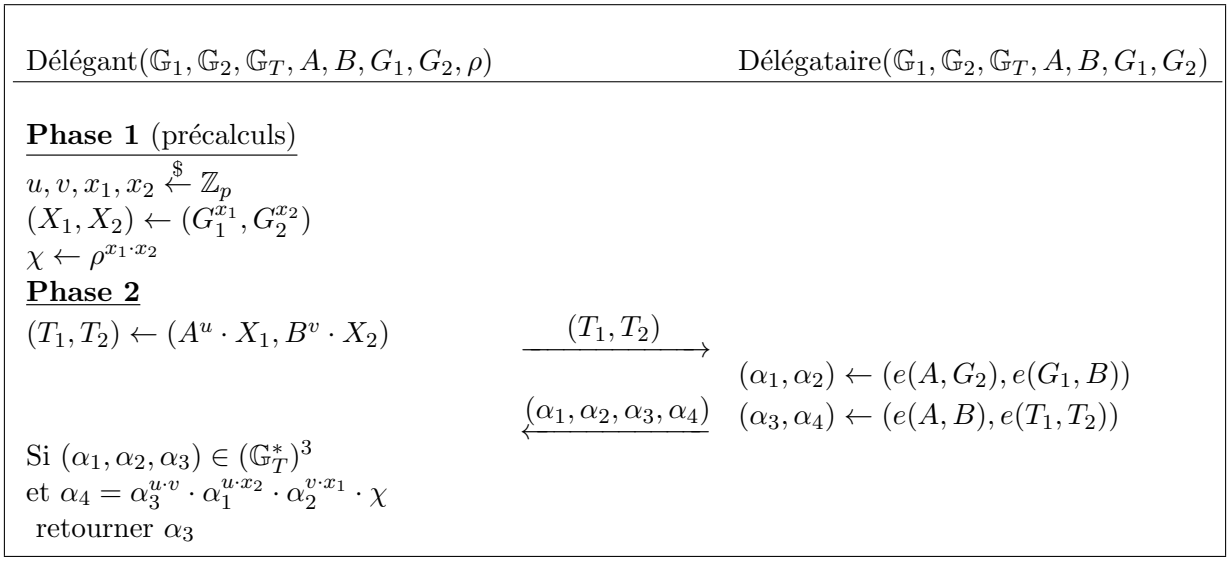


Figure 7.4 – Protocole de délégation du couplage $e(A, B)$ de Chevallier-Mames *et al*, où $A \in \mathbb{G}_1^*$ et $B \in \mathbb{G}_2^*$ sont des points publics.

discutons de l'intérêt et du coût de ces tests d'appartenance dans la section 7.3.3 ci-dessous. On peut noter que Chevallier-Mames *et al.* proposèrent dans le même article une variante de ce protocole assurant également la confidentialité. Nous en rappelons l'efficacité dans le tableau 7.1.

7.3.2 Un Nouveau Protocole

Les articles [CCM+05, KLP05, CCM+10] suivent la même approche. Ils commencent par construire un protocole vérifiable $\mathcal{P}'$ pour des points secrets A et B qu'ils vont ensuite chercher à décliner en un protocole vérifiable $\mathcal{P}$ pour des points publics.

L'approche inverse semble pourtant plus intéressante en raison des travaux de Girault et Lefranc. En effet, comme nous l'avons expliqué dans [CDS14], $\mathcal{P}'$ peut très simplement être construit à partir de n'importe quel protocole $\mathcal{P}$. Il suffit simplement d'exécuter $\mathcal{P}$ sur $U \leftarrow A^u$ et $V \leftarrow B^v$ (où u et v sont des scalaires aléatoires) et d'élever le résultat à la puissance $(u \cdot v)^{-1}$. Le protocole ainsi obtenu vérifie clairement la propriété de confidentialité, et ce, pour un surcoût raisonnable. En particulier, il est plus efficace d'utiliser cette méthode pour rendre confidentiel le protocole de la figure 7.4 que d'utiliser sa variante pour points secrets proposée dans [CCM+05].

Dans cette section nous nous concentrons donc uniquement sur la conception d'un protocole vérifiable. Comme l'ensemble des protocoles assurant cette propriété face à un délégataire unique, l'idée de notre construction est de demander à ce dernier le calcul d'une valeur secrètement reliée au couplage $e(A, B)$. L'apport de notre solution est de définir une relation qui est à la fois simple à vérifier et difficile à deviner pour le délégataire. Elle offre donc une meilleure efficacité sans affaiblir la sécurité.

La figure 7.5 décrit notre protocole de délégation. Celui-ci suppose, comme précédemment, la connaissance d'un couplage $\rho \leftarrow e(G_1, G_2)$ pour des points publics G_1 et G_2.

Complétude. L'exécution de ce protocole entre deux entités honnêtes permet au délégant d'ob-

SIM$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, A, B, G_1, G_2, \rho)$		Téléphone$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, A, B, G_1, G_2)$
Phase 1 (précalculs)		
$x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$		
$(X_1, X_2) \leftarrow (G_1^{x_1}, G_2^{x_2})$		
$\chi \leftarrow \rho^{x_1 \cdot x_2}$		
Phase 2		
$(T_1, T_2) \leftarrow (A^{x_2^{-1}} \cdot X_1, B^{x_1^{-1}} \cdot X_2)$	$\xrightarrow{(T_1, T_2)}$	
		$\alpha_1 \leftarrow e(T_1, T_2)[(e(A, G_2) \cdot e(G_1, B))]^{-1}$
	$\xleftarrow{(\alpha_1, \alpha_2)}$	$\alpha_2 \leftarrow e(A, B)$
Si $\alpha_2 \in \mathbb{G}_T^*$		
et $\alpha_1 = \chi \cdot \alpha_2^{(x_1 \cdot x_2)^{-1}}$		
retourner α_2		

Figure 7.5 – Protocole de délégation du couplage $e(A, B)$, présenté dans [CDS14], où $A \in \mathbb{G}_1^*$ et $B \in \mathbb{G}_2^*$ sont des points publics.

tenir la valeur de $e(A, B)$. En effet :

$$\begin{aligned}
\alpha_1 &= e(T_1, T_2)[(e(A, G_2) \cdot e(G_1, B))]^{-1} \\
&= e(A^{x_2^{-1}} \cdot X_1, B^{x_1^{-1}} \cdot X_2)[(e(A, G_2) \cdot e(G_1, B))]^{-1} \\
&= e(A, B)^{(x_1 \cdot x_2)^{-1}} \cdot e(X_1, X_2) \\
&= \chi \cdot \alpha_2^{(x_1 \cdot x_2)^{-1}}
\end{aligned}$$

Vérifiabilité. L'objectif d'un délégataire malhonnête est de retourner une mauvaise valeur de $e(A, B)$ qui sera acceptée par le délégant. Pour cela, il doit retourner $\alpha_1' = \alpha_1 \cdot \gamma$ et $\alpha_2' = \alpha_2 \cdot \delta$, avec $(\gamma, \delta) \in \mathbb{G}_T \times \mathbb{G}_T^*$ (car l'appartenance à ce groupe est vérifiée dans le protocole), tels que :

$$\alpha_1' = \chi \cdot (\alpha_2')^{(x_1 \cdot x_2)^{-1}}$$

Par conséquent, $\alpha_1 \cdot \gamma = \chi \cdot (\alpha_2 \cdot \delta)^{(x_1 \cdot x_2)^{-1}}$ et donc $\gamma^{(x_1 \cdot x_2)} = \delta$. Casser la propriété de vérifiabilité revient donc à trouver deux éléments γ et δ de $\mathbb{G}_T^*$ satisfaisant cette dernière équation. Malheureusement, cela ne correspond à aucune hypothèse calculatoire connue, il nous faut donc étudier la difficulté de ce problème dans le modèle des groupes génériques introduit par Shoup [Sho97]. Celui-ci représente les éléments des différents groupes par des chaînes de bits arbitraires sur lesquelles l'adversaire ne peut effectuer aucun autre test que celui d'égalité. Cela implique notamment que les opérations entre les différents éléments ne peuvent s'effectuer qu'au travers de requêtes à des oracles spécifiques. Dans le cas des groupes bilinéaires, l'adversaire a donc accès à 3 oracles gérant respectivement les opérations dans $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$, ainsi qu'à un oracle lui permettant d'obtenir le résultat du couplage de deux points.

Pour plus de simplicité, nous considérons que les groupes bilinéaires $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ impliqués sont de type 3, ce qui correspond à la très grande majorité des cas en pratique. Il est néanmoins possible d'étendre cette étude aux autres types de couplages mais il y a alors beaucoup plus de combinaisons à prendre en compte.

Soient a, b, x_1, x_2 tels que $A = G_1^a$, $B = G_2^b$, $X_1 = G_1^{x_1}$ et $X_2 = G_2^{x_2}$. Afin d'obtenir les meilleures garanties de sécurité, nous considérons que l'adversaire peut choisir les points A et B,

et qu'il connait donc les valeurs a et b. Nous associons, dans ce qui suit, les éléments des différents groupes à des polynômes dont les variables sont les inconnues $x_1, x_1^{-1}, x_2, x_2^{-1}$. Nous commençons par prouver qu'il est impossible, en combinant formellement les éléments connus de l'adversaire, de produire une paire $(\gamma, \delta) \in (\mathbb{G}_T^*)^2$ permettant de tromper le délégant, c'est-à-dire telle que $\delta = \gamma^{(x_1 \cdot x_2)}$.

Dans le modèle des groupes génériques, les éléments de $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) connus de l'adversaire ne peuvent être que des combinaisons de G_1 et T_1 (resp. G_2 et T_2). Par conséquent, les seuls éléments de $\mathbb{G}_T$ auxquels il a accès sont de la forme $e(G_1^{a_1} \cdot T_1^{a_2}, G_2^{a_3} \cdot T_2^{a_4})$ pour des scalaires $a_1, a_2, a_3, a_4 \in \mathbb{Z}_p$. On peut noter que cette formule regroupe également les combinaisons avec les points A et B (car l'adversaire connait a et b) ainsi qu'avec la valeur $\rho = e(G_1, G_2)$. Les éléments γ et δ utilisés par l'adversaire s'écrivent donc :

$$\gamma = e(G_1^{a_1} \cdot T_1^{a_2}, G_2^{a_3} \cdot T_2^{a_4}) \text{ et } \delta = e(G_1^{a'_1} \cdot T_1^{a'_2}, G_2^{a'_3} \cdot T_2^{a'_4})$$

pour des scalaires $(a_i, a'_i)_{i=1}^4 \in \mathbb{Z}_p^8$. En remplaçant T_1 par $A^{x_2^{-1}} \cdot X_1 = G_1^{a \cdot x_2^{-1} + x_1}$ et T_2 par $B^{x_1^{-1}} \cdot X_2 = G_2^{a \cdot x_1^{-1} + x_2}$, on obtient que $\gamma = \rho^r$ et $\delta = \rho^z$ où :

$$r = (a_2a_4ab)x_1^{-1}x_2^{-1} + (a_1a_4b)x_1^{-1} + (a_2a_3a)x_2^{-1} + a_1a_3 + a_2a_4(a+b) + (a_2a_3)x_1 + (a_1a_4)x_2 + (a_2a_4)x_1x_2$$

$$\text{et } z = (a'_2a'_4ab)x_1^{-1}x_2^{-1} + (a'_1a'_4b)x_1^{-1} + (a'_2a'_3a)x_2^{-1} + a'_1a'_3 + a'_2a'_4(a+b) + (a'_2a'_3)x_1 + (a'_1a'_4)x_2 + (a'_2a'_4)x_1x_2$$

La relation $\delta = \gamma^{x_1x_2}$ impose alors les restrictions suivantes :

- $a_2a_3 = 0$;
- $a_1a_4 = 0$;
- $a_2a_4 = 0$.

En effet z ne contient aucun terme en $x_1^2x_2$, $x_1x_2^2$ ou $x_1^2x_2^2$. Par conséquent, on a $r = a_1a_3 \neq 0$ (car $\delta \in \mathbb{G}_T^*$) et donc $z = (a_1a_3)x_1x_2$, ce qui permet de déduire que $a'_2a'_4 = a_1a_3 \neq 0$. Cela entraîne une contradiction sur le coefficient $a'_2a'_4ab$ du monôme $x_1^{-1}x_2^{-1}$ qui doit être nul en raison de l'écriture $z = (a_1a_3)x_1x_2$. Comme $a \neq 0$ et $b \neq 0$, on a $0 = a'_2a'_4 = a_1a_3 \neq 0$, ce qui est impossible. L'adversaire est donc incapable de produire formellement une paire (γ, δ) permettant de tromper le délégant.

Il ne reste alors qu'à estimer la probabilité que deux polynômes aient la même image pour des valeurs $x_1, x_2 \in \mathbb{Z}_p$. Soit q_G une borne sur le nombre de requêtes aux oracles d'opérations de groupe ou de couplage. Le nombre maximal de polynômes connus de l'adversaire est alors majoré par $4 + q_G$ (car il connait également G_1, G_2, T_1, T_2). Par ailleurs, chacun d'entre eux est de degré au plus 2. Par conséquent, le lemme de Schwartz-Zippel affirme que la probabilité que deux de ces polynômes atteignent la même image est majorée par $(4 + q_G)^2/p$, ce qui est négligeable.

Dans le modèle des groupes génériques, notre protocole est donc vérifiable avec une probabilité à distance négligeable de 1.

7.3.3 Test d'Appartenance

Notre protocole, de même que celui de la figure 7.4, impose au délégant de tester l'appartenance au groupe $\mathbb{G}_T$ de certains éléments retournés par le délégataire. En effet, nous rappelons (voir section 2.2.2) que ce groupe est inclus dans $\mathbb{F}_{q^k}^*$, où q est la taille du corps sur lequel est défini la courbe elliptique et k est le degré de plongement. Le risque est donc que l'adversaire se serve d'éléments d'ordres petits pour optimiser ses chances de succès. Par exemple, si celui-ci retourne α_1 et $\alpha'_2 = \alpha_2 \cdot \mu$ où μ est l'élément de $\mathbb{F}_{q^k}^*$ d'ordre 2, il y a alors 1 chance sur 2 que l'équation $\alpha_1 = \chi \cdot (\alpha'_2)^{(x_1 \cdot x_2)^{-1}}$ soit vérifiée. Plus généralement, si μ est d'ordre r, l'adversaire pourra casser la vérifiabilité avec une probabilité de $1/r$.

Intuitivement, l'objectif du test d'appartenance au groupe $\mathbb{G}_T$ est donc d'empêcher l'adversaire de retourner des éléments dont l'ordre admet des petits facteurs. Pour éviter une coûteuse

élévation à la puissance p dans $\mathbb{F}_{q^k}^*$, Scott [Sco13] proposa de tester plutôt l'appartenance au groupe cyclotomique $\mathbb{G}_\phi \supset \mathbb{G}_T$. L'intérêt de ce groupe est qu'il est d'ordre $\phi_k(q)$, où ϕ_k est le k-ième polynôme cyclotomique. Le test d'appartenance consiste donc essentiellement à appliquer le morphisme de Frobenius, ce qui a un coût négligeable pour des éléments de $\mathbb{F}_{q^k}^*$. Néanmoins, pour que ce test soit utile, il faut que $\phi_k(q)$ n'ait pas de petits facteurs et s'écrive donc $p \cdot \ell$, où ℓ est un produit de grands nombres premiers.

Cette solution nécessite par conséquent un choix approprié des paramètres de la courbe, ce qui n'est pas toujours possible. Nous avons donc proposé dans [CDS14] une alternative pour les autres cas. Celle-ci permet de fusionner le test d'appartenance à $\mathbb{G}_T$ (qui nécessite une exponentiation dans $\mathbb{F}_{p^k}^*$) et l'équation de vérification (qui nécessite une exponentiation dans $\mathbb{G}_T$) en une seule exponentiation dans $\mathbb{F}_{p^k}^*$. Elle repose sur le fait que les couplages utilisés en pratique sont issus du couplage de Tate et peuvent donc être divisés en deux parties appelées algorithme de Miller et exponentiation finale.

Soit $c = \frac{q^k-1}{p}$ l'exposant intervenant dans cette dernière phase. Pour tout élément $\alpha \in \mathbb{G}_T$, nous désignons par $\widetilde{\alpha}$ un élément de $\mathbb{F}_{q^k}^*$ tel que $(\widetilde{\alpha})^c = \alpha$ (concrètement $\widetilde{\alpha}$ est la sortie de l'algorithme de Miller). Supposons que le délégant calcule désormais $\widetilde{\chi} \leftarrow (\widetilde{\rho})^{x_1 \cdot x_2}$ ($\widetilde{\rho}$ serait contenu dans les paramètres du système) et que le délégataire retourne $\widetilde{\alpha_1}$ au lieu de α_1. Il suffit alors au délégant de vérifier que l'équation suivante est satisfaite :

$$\alpha_2 \stackrel{?}{=} (\widetilde{\alpha_1} \cdot \rho^{-1})^{c \cdot x_1 \cdot x_2}.$$

Puisque α_2 est égal à un élément de $\mathbb{F}_{q^k}^*$ élevé à la puissance c, il est d'ordre p et donc dans $\mathbb{G}_T$. Le test d'appartenance n'est alors plus nécessaire.

7.3.4 Efficacité

	Conf.	Vérif.	Phase 1	Phase 2	Délégat.	Rapport
[GL05]	oui	non	-	$1\mathtt{E}_{\mathbb{G}_1}+1\mathtt{E}_{\mathbb{G}_2}+1\mathtt{E}_{\mathbb{G}_T}$	1P	0.46
[CCM+05] [Sect 4.1]	oui	oui	$2\mathtt{E}_{\mathbb{G}_1}+2\mathtt{E}_{\mathbb{G}_2}+2\mathtt{E}_{\mathbb{G}_T}$	$1\mathtt{E}_{\mathbb{G}_1}+1\mathtt{E}_{\mathbb{G}_2}+5\mathtt{E}_{\mathbb{G}_T}+3\mathtt{T}_T$	4P	1.46
[CCM+05] [Sect 5.2]	non	oui	$1\mathtt{E}_{\mathbb{G}_1}+1\mathtt{E}_{\mathbb{G}_2}+1\mathtt{E}_{\mathbb{G}_T}$	$1\mathtt{E}_{\mathbb{G}_1}+1\mathtt{E}_{\mathbb{G}_2}+3\mathtt{E}_{\mathbb{G}_T}+3\mathtt{T}_T$	4P	0.96
[KLP05]	oui	faible	$1\mathtt{E}_{\mathbb{G}_1}+1\mathtt{E}_{\mathbb{G}_2}+1\mathtt{E}_{\mathbb{G}_T}$	$1\mathtt{E}_{\mathbb{G}_1}+1\mathtt{E}_{\mathbb{G}_2}+3\mathtt{E}_{\mathbb{G}_T}+3\mathtt{T}_T$	4P	0.96
Nous	non	oui	$1\mathtt{E}_{\mathbb{G}_1}+1\mathtt{E}_{\mathbb{G}_2}+1\mathtt{E}_{\mathbb{G}_T}$	$1\mathtt{E}_{\mathbb{G}_1}+1\mathtt{E}_{\mathbb{G}_2}+1\mathtt{E}_{\mathbb{G}_T}+1\mathtt{T}_T$	4P	0.46
Nous + [GL05]	oui	oui	$1\mathtt{E}_{\mathbb{G}_1}+1\mathtt{E}_{\mathbb{G}_2}+1\mathtt{E}_{\mathbb{G}_T}$	$2\mathtt{E}_{\mathbb{G}_1}+2\mathtt{E}_{\mathbb{G}_2}+2\mathtt{E}_{\mathbb{G}_T}+1\mathtt{T}_T$	4P	0.92

Table 7.1 – Comparaison des performances de notre protocole avec celles de l'état de l'art. La colonne « Conf. » (resp. « Vérif. ») indique le caractère confidentiel (resp. vérifiable) du protocole. La vérifiabilité « faible » fait référence à celle satisfaite par le protocole de Kang *et al.* (voir début de la section 7.3). La notation $\mathtt{E}_{\mathbb{G}_1}$ (resp. $\mathtt{E}_{\mathbb{G}_2}$) désigne une exponentiation dans $\mathbb{G}_1$ (resp. $\mathbb{G}_2$), $\mathtt{E}_{\mathbb{G}_T}$ une exponentiation dans $\mathbb{G}_T$, $\mathtt{P}$ un couplage et $\mathtt{T}_T$ un test d'appartenance au groupe $\mathbb{G}_T$. La colonne « Rapport » fait l'hypothèse que la méthode de Scott s'applique pour le test d'appartenance au groupe $\mathbb{G}_T$ (voir section 7.3.3). La dernière ligne combine notre protocole avec celui de Girault et Lefranc afin d'obtenir une version confidentielle et vérifiable (voir section 7.3.2).

Nous comparons, dans le tableau 7.1, l'efficacité des protocoles de délégation de couplages

présentées dans ce chapitre. Leur complexité est représentée par le nombre d'exponentiations dans $\mathbb{G}_1$, $\mathbb{G}_2$ ou $\mathbb{G}_T$ qu'ils nécessitent. Il est important de noter que le gain par rapport à l'exécution d'un couplage dépend totalement du choix de la courbe elliptique et peut, dans certains cas, être très faible. Il existe cependant des paramètres pour lesquels ce gain est significatif. Afin de l'illustrer, nous donnons, dans le cas des courbes KSS-18 [KSS08], le rapport entre le coût de la phase 2 des différents protocoles et le coût d'un couplage. Celui-ci a été obtenu à partir des temps d'exécution fournis dans l'article [BCN14].

Chapitre 8

Conception d'un schéma de signature polyvalent et efficace

Sommaire

Nous présentons dans ce chapitre un nouveau schéma de signature numérique ainsi que plusieurs de ses applications. L'essentiel des résultats présentés ici est issu de l'article *Short Randomizable Signature* [PS15], cosigné avec David Pointcheval.

8.1 Une Nouvelle Construction de Signatures Courtes

Comme nous l'avons vu dans ce mémoire, les signatures numériques sont souvent utilisées comme briques de base pour des constructions plus complexes, telles que les signatures de groupe, les attestations anonymes mais également la monnaie électronique. Une situation que rencontre fréquemment l'utilisateur de ces systèmes est celle où il obtient une signature σ sur un message secret s et doit ensuite prouver, à plusieurs reprises, que s est certifié, tout en étant non-traçable. L'utilisateur, ne pouvant bien évidemment pas révéler son secret, va devoir prouver la connaissance de s et montrer que celui-ci a été signé. Il est donc indispensable que le schéma de signature et le système de preuves soient compatibles.

En pratique, dans un environnement bilinéaire, les protocoles cryptographiques utilisent soit les preuves Groth-Sahai, soit des preuves rendues non interactives à l'aide de la méthode Fiat-Shamir. Dans le premier cas, il sera nécessaire d'avoir recours à un schéma de signature *conservant la structure* [AFG+10], c'est-à-dire un schéma où les messages et les signatures appartiennent tous aux groupes $\mathbb{G}_1$ ou $\mathbb{G}_2$. En effet, les preuves Groth-Sahai ne permettent pas de prouver la connaissance de scalaires mais seulement d'éléments de ces groupes. Dans le deuxième cas, on préfèrera au contraire un schéma de signature signant des scalaires car cela permettra de choisir

s comme le logarithme discret d'un certain élément public et donc d'utiliser une preuve de type Schnorr.

Bien que les preuves Fiat-Shamir n'offrent pas les mêmes garanties de sécurité que les preuves Groth-Sahai elles permettent d'atteindre une efficacité incomparable avec ces dernières et sont donc privilégiées en pratique. Il existe alors essentiellement deux choix possibles pour instancier le schéma de signature : les signatures Boneh-Boyen (BB) et les signatures Camenisch-Lysyanskaya (CL) (*cf.* partie 3.2.2). A priori, la comparaison semble être nettement à l'avantage des premières dont une extension présentée dans [ASM06] permet d'obtenir des signatures de taille constante sur plusieurs messages. Cependant, le choix n'est pas aussi évident car les signatures CL proposent plusieurs fonctionnalités, telles que leur capacité à être régénérées, permettant d'améliorer significativement l'efficacité des constructions les utilisant. En effet, reprenons l'exemple décrit dans le premier paragraphe de ce chapitre. Si σ est une signature BB, alors l'utilisateur ne peut la révéler pour prouver que s est certifié car celle-ci permettrait de le tracer. Il doit alors prouver sa connaissance de σ et de s et montrer que le premier est bien une signature sur le deuxième, ce qui complexifie la preuve à divulgation nulle. Supposons maintenant que σ soit une signature CL de la forme $(a, b, c) \in \mathbb{G}_1^3$. A chaque fois que l'utilisateur souhaitera prouver que s est certifié il lui suffira de régénérer σ en calculant $\sigma' \leftarrow (a^t, b^t, c^t)$, de fournir σ' et de montrer que c'est une signature valide sur s. La preuve à divulgation nulle est alors plus simple car il y a moins d'éléments dont il faut prouver la connaissance. La non-traçabilité reposera ici sur la difficulté de relier σ et σ', ce qui correspond au problème XDH. Ainsi, la perte d'efficacité entraînée par le choix des signatures CL peut être compensée par la possibilité de disposer de preuves à divulgation nulle plus simples. C'est ce qui explique que les schémas de signature de groupe et d'attestation anonyme les plus efficaces utilisent aujourd'hui les signatures CL.

Cependant, l'intérêt des signatures CL diminue rapidement lorsque le nombre de messages à signer augmente. En effet, celles-ci souffrent d'une taille linéaire en le nombre de messages qui peut rapidement devenir prohibitive pour certaines applications. C'est par exemple le cas des accréditations anonymes où les utilisateurs ont en général plusieurs attributs à faire certifier. En pratique ces systèmes [ASM06, CL13, BL13] doivent donc utiliser d'autres schémas de signature (le plus souvent celui de Boneh Boyen), ce qui est regrettable car, là encore, les propriétés des signatures CL pourraient les rendre plus efficaces.

Dans cette section, nous présentons un nouveau schéma de signature corrigeant les problèmes de complexité des signatures CL. En effet, nos signatures proposent les mêmes fonctionnalités que ces dernières mais bénéficient d'une taille constante (c'est-à-dire ne dépendant pas du nombre de messages signés) et d'algorithmes plus efficaces. Notre construction, qui se compare favorablement aux autres schémas de l'état de l'art, se révèle intéressante tant pour un usage classique (c'est-à-dire signer un document) que pour servir de briques de base dans des protocoles cryptographiques plus complexes. En effet, dans le premier cas, la capacité de nos signatures à s'agréger (voir section 8.2) se révèlera utile à plusieurs cas d'usage tandis que dans le deuxième cas, elles permettront (voir section 8.3) d'améliorer l'efficacité de nombreux protocoles, les rendant ainsi plus accessibles pour des utilisations concrètes.

8.1.1 Description du Schéma

Notre construction part du constat que les signatures CL, comme la majorité des protocoles du début de la cryptographie à base de couplage, ont été conçues pour des groupes bilinéaires de type 1. Pourtant, la plupart des protocoles récents qui les utilisent [BCN+10, CPS10, BFG+13] nécessitent des groupes bilinéaires de type 3 pour des raisons d'efficacité ou de sécurité. Il existe, certes, une version asymétrique des signatures CL (décrite dans la section 3.2.2) mais celle-ci n'est qu'une simple transposition de la version symétrique. Ces dernières années, des travaux [AGHO11, CM14] sur les signatures conservant la structure ont prouvé que concevoir des schémas spécifiquement pour les groupes de type 3 permettait d'en améliorer sensiblement l'efficacité. En

appliquant la même idée aux signatures CL nous proposons un nouveau schéma tirant parti de tout le potentiel de ces groupes.

Afin de mieux présenter l'intuition de notre construction, nous commençons par décrire le cas particulier où un seul message est signé avant d'en présenter une extension permettant de signer plusieurs messages à la fois.

Protocole pour signer un message

- $\texttt{Setup}(1^k)$: soit k le paramètre de sécurité, cet algorithme retourne la description $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ de groupes bilinéaires de type 3.
- $\texttt{Keygen}(p.p.)$: cet algorithme génère $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2$ et deux scalaires $(x, y) \xleftarrow{\$} \mathbb{Z}_p^2$ puis calcule $(\widetilde{X}, \widetilde{Y}) \leftarrow (\widetilde{g}^x, \widetilde{g}^y)$. La clé secrète sk est alors définie comme étant (x, y) et la clé publique pk comme étant $(\widetilde{g}, \widetilde{X}, \widetilde{Y})$.
- $\texttt{Sign}(\mathsf{sk}, m)$: pour signer un message m, cet algorithme sélectionne un générateur aléatoire $h \xleftarrow{\$} \mathbb{G}_1^*$ et retourne $\sigma \leftarrow (h, h^{(x+y\cdot m)})$.
- $\texttt{Verify}(\mathsf{pk}, \sigma, m)$: étant donnée une signature $\sigma = (\sigma_1, \sigma_2)$, l'algorithme vérifie si les conditions $\sigma_1 \neq 1_{\mathbb{G}_1}$ et $e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^m) = e(\sigma_2, \widetilde{g})$ sont satisfaites. Il retourne 1 si c'est le cas et 0 sinon.

Complétude : Si $\sigma = (\sigma_1 = h, \sigma_2 = h^{(x+y\cdot m)})$, alors

$$e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^m) = e(h, \widetilde{X} \cdot \widetilde{Y}^m) = e(h, \widetilde{g})^{(x+y\cdot m)} = e(h^{(x+y\cdot m)}, \widetilde{g}) = e(\sigma_2, \widetilde{g}).$$

Remarque 23. Il est également possible de définir la clé secrète sk comme étant (g, X, Y) où g serait un générateur de $\mathbb{G}_1$ et $(X, Y) \leftarrow (g^x, g^y)$. Une signature σ sur un message m serait alors générée en calculant $(g^r, (X \cdot Y^m)^r)$ pour un certain scalaire $r \xleftarrow{\$} \mathbb{Z}_p$. Cependant, cette solution impliquerait trois exponentiations dans le calcul d'une signature contre une seule lorsque (x, y) est connu.

Intuitivement, l'efficacité de notre construction repose sur la séparation entre $\mathbb{G}_1$, qui est l'espace des signatures et $\mathbb{G}_2$, qui est l'espace de la clé publique pk. En effet, si nous avions utilisé des couplages symétriques, la clé publique pk deviendrait le triplet $(g, X, Y) \in \mathbb{G}_1^3$ qui permet de signer. Il faudrait alors construire notre signature de manière plus complexe, comme c'est le cas pour les signatures CL. L'absence d'isomorphisme calculable entre $\mathbb{G}_1$ et $\mathbb{G}_2$ permet ici de garantir que les éléments de pk ne seront pas utilisés pour falsifier les signatures et donc de proposer des constructions plus simples.

Le schéma décrit ci-dessus peut facilement être modifié pour permettre de signer plusieurs messages à la fois. Le point important est que les signatures ainsi obtenues restent de taille constante, consistant en seulement deux éléments de $\mathbb{G}_1$.

Protocole pour signer r messages

- $\texttt{Setup}(1^k)$: soit k le paramètre de sécurité, cet algorithme retourne la description $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ de groupes bilinéaires de type 3.
- $\texttt{Keygen}(p.p.)$: cet algorithme génère $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2$ et $(x, y_1, \ldots, y_r) \xleftarrow{\$} \mathbb{Z}_p^{r+1}$ puis calcule $(\widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r) \leftarrow (\widetilde{g}^x, \widetilde{g}^{y_1}, \ldots, \widetilde{g}^{y_r})$. La clé secrète sk est alors définie comme étant $(x, y_1, \ldots, y_r)$ et la clé publique pk comme étant $(\widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r)$.
- $\texttt{Sign}(\mathsf{sk}, m_1, \ldots, m_r)$: pour signer les messages $m_1, \ldots m_r$, cet algorithme sélectionne un générateur aléatoire $h \xleftarrow{\$} \mathbb{G}_1^*$ et retourne $\sigma \leftarrow (h, h^{(x+\sum y_j \cdot m_j)})$.
- $\texttt{Verify}(\mathsf{pk}, (m_1, \ldots, m_r), \sigma)$: étant donnée une signature $\sigma = (\sigma_1, \sigma_2)$, l'algorithme vérifie si les conditions $\sigma_1 \neq 1_{\mathbb{G}_1}$ et $e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j}) = e(\sigma_2, \widetilde{g})$ sont satisfaites. Il retourne 1 si c'est le cas et 0 sinon.

Complétude : Si $\sigma = (\sigma_1 = h, \sigma_2 = h^{(x+\sum y_j \cdot m_j)})$, alors

$$e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j}) = e(h, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j}) = e(h, \widetilde{g})^{x+\sum y_j \cdot m_j}$$
$$= e(h^{x+\sum y_j \cdot m_j}, \widetilde{g}) = e(\sigma_2, \widetilde{g}).$$

Remarque 24. Tout comme les signatures CL, nos signatures peuvent être régénérées. En effet, si $\sigma = (\sigma_1, \sigma_2)$ est une signature valide sur le vecteur $(m_1, \ldots, m_r)$, il en est de même pour $\sigma' \leftarrow (\sigma_1^t, \sigma_2^t)\ \forall t \in \mathbb{Z}_p$. Cela correspond simplement à remplacer, lors de la génération de la signature, h par h^t. Plus généralement, nos signatures offrent les mêmes fonctionnalités que les signatures CL et peuvent donc facilement les remplacer, comme nous l'illustrons dans les sections 8.2 et 8.3.

Nous comparons dans le tableau 8.2 l'efficacité de notre construction avec celles des deux principaux schémas de l'état de l'art. Le point fort de notre schéma est qu'il concilie les avantages des signatures BB, à savoir une taille constante et des algorithmes très efficaces et celles des signatures CL. L'extension des signatures BB décrites dans [ASM06], permettant de signer r messages, considère des signatures de la forme (A, e, s) où e et s sont des scalaires aléatoires et $A \leftarrow (g_0 g_1^s g_2^{m_1} \ldots g_{r+1}^{m_r})^{\frac{1}{\gamma+e}}$ pour des paramètres publics $g_0, \ldots g_{r+1} \in \mathbb{G}_1$ et une clé secrète $\gamma \in \mathbb{Z}_p$. Par souci d'équité, nous considérons une version plus efficace où le signataire connait les logarithmes discrets y_i des éléments g_i en base g. Ainsi, il peut calculer A en seulement une exponentiation : $A \leftarrow g^{\frac{1+\sum_{i=1}^{r+1} y_i \cdot m_i}{\gamma+e}}$

	Taille	Coût (Sign.)	Coût (Vérif.)	Régén.	Groupes
BB [ASM06]	$1\,\mathbb{G}_1 + 2\,\mathbb{Z}_p$	$2\,\mathtt{R}_{\mathbb{Z}_p} + 1\,\mathtt{E}_{\mathbb{G}_1}$	$2\,\mathtt{P} + 1\,\mathtt{E}_{\mathbb{G}_2} + (r+1)\,\mathtt{E}_{\mathbb{G}_1}$	Non	Tous
CL [CL04]	$(1+2r)\,\mathbb{G}_1$	$1\,\mathtt{R}_{\mathbb{G}_1} + 2r\,\mathtt{E}_{\mathbb{G}_1}$	$4r\,\mathtt{P} + r\,\mathtt{E}_{\mathbb{G}_2}$	Oui	Tous
Nous	$2\,\mathbb{G}_1$	$1\,\mathtt{R}_{\mathbb{G}_1} + 1\,\mathtt{E}_{\mathbb{G}_1}$	$2\,\mathtt{P} + r\,\mathtt{E}_{\mathbb{G}_2}$	Oui	Type 3

Table 8.1 – Comparaison entre notre schéma de signature et l'état de l'art. Ici, r correspond au nombre de messages signés, $\mathtt{Rand}_{\mathbb{G}_1}$ (resp. $\mathtt{Rand}_{\mathbb{Z}_p}$) au coût de la génération d'un élément aléatoire de $\mathbb{G}_1$ (resp. $\mathbb{Z}_p$), $\mathtt{exp}_{\mathbb{G}_i}$ au coût d'une exponentiation dans $\mathbb{G}_i$ ($i \in \{1, 2\}$) et $\mathtt{P}$ au coût d'un calcul de couplage.

8.1.2 Étude de la Sécurité

Tout comme les signatures BB et CL lorsqu'elles furent proposées, notre construction ne correspond à aucune hypothèse de sécurité existante. Nous introduisons une nouvelle hypothèse dont nous étudions la robustesse dans le modèle des groupes génériques proposés par Shoup [Sho97]. Bien sûr, une preuve dans ce modèle n'offre pas les mêmes garanties de sécurité qu'une réduction à un problème bien connu mais nous soutenons que les gains de performance permis par nos signatures peuvent justifier le recours à cette nouvelle hypothèse.

Définition 46 (Hypothèse 1)**.** Soient $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires de type 3 et g (resp. $\widetilde{g}$) un générateur de $\mathbb{G}_1$ (resp. $\mathbb{G}_2$). Soient $\widetilde{X} = \widetilde{g}^x$ et $\widetilde{Y} = \widetilde{g}^y$ pour des scalaires aléatoires x et y, nous définissons l'oracle $\mathcal{O}(m)$, qui prend en entrée $m \in \mathbb{Z}_p$ et retourne une paire (h, h^{x+ym}) pour un certain générateur aléatoire h de $\mathbb{G}_1$. Étant donnés $(\widetilde{g}, \widetilde{X}, \widetilde{Y})$ et un accès illimité à cet oracle, aucun adversaire ne peut générer une paire similaire avec $h \neq 1_{\mathbb{G}_1}$ pour un scalaire m^* qui n'a pas été soumis à $\mathcal{O}$.

La sécurité EUF-CMA de notre schéma pour signer un message correspond exactement à cette nouvelle hypothèse, l'oracle de signature décrit dans la figure 3.1 de la section 3 étant équivalent

à $\mathcal{O}$. Pour pouvoir signer des messages mis en gage, nous aurons cependant besoin d'une version un peu plus forte de cette hypothèse que nous décrivons ci-dessous.

Définition 47 (Hypothèse 2)**.** Soient $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ la description de groupes bilinéaires de type 3 et g (resp. $\widetilde{g}$) un générateur de $\mathbb{G}_1$ (resp. $\mathbb{G}_2$). Soient $(X = g^x, Y = g^y)$ et $(\widetilde{X} = \widetilde{g}^x, \widetilde{Y} = \widetilde{g}^y)$ pour des scalaires aléatoires x et y, nous définissons l'oracle $\mathcal{O}(m)$, qui prend en entrée $m \in \mathbb{Z}_p$ et retourne une paire (h, h^{x+ym}) pour un certain générateur aléatoire h de $\mathbb{G}_1$. Étant donnés $(g, Y, \widetilde{g}, \widetilde{X}, \widetilde{Y})$ et un accès illimité à cet oracle, aucun adversaire ne peut générer une paire similaire avec $h \neq 1_{\mathbb{G}_1}$ pour un scalaire m^* qui n'a pas été soumis à $\mathcal{O}$.

L'adversaire de cette nouvelle hypothèse a donc accès à davantage d'éléments, à savoir $g, Y \in \mathbb{G}_1$. Nous montrons cependant que celle-ci reste valide dans le modèle des groupes génériques.

Théorème 48. *L'hypothèse 2 (et donc l'hypothèse 1) décrite ci-dessus est valide dans le modèle des groupes génériques : aucun adversaire, après q requêtes à l'oracle $\mathcal{O}$ et q_G requêtes aux oracles d'opérations de groupe, ne peut générer une paire valide avec une probabilité supérieure à $3(5 + 2q + q_G)^2/2p$.*

Preuve. Soient g (resp. $\widetilde{g}$) un générateur de $\mathbb{G}_1$ (resp. $\mathbb{G}_2$), x et y les scalaires définissant (X, Y) et $(\widetilde{X}, \widetilde{Y})$, et $r_i \in \mathbb{Z}_p^*$ le scalaire tel que la paire retournée par $\mathcal{O}$ lors de la i-ème requête est $(h_i = g^{r_i}, t_i = h_i^{x+y \cdot m_i})$ (où m_i est le message soumis). Nous associons, dans ce qui suit, les éléments des différents groupes à des polynômes dont les variables sont les inconnues $x, y, r_1, \ldots, r_q$. Nous devons alors prouver qu'il est impossible, en combinant formellement des éléments connus de l'adversaire, de produire une nouvelle paire valide. Nous devons également estimer la probabilité que deux polynômes retournés par les différents oracles atteignent, une fois évalués, la même image.

Soit (h^*, t^*) la paire retournée par un adversaire sur un scalaire m^*. Puisque h^* et t^* sont des éléments de $\mathbb{G}_1$, ils ne peuvent être que des combinaisons des précédentes paires (h_i, t_i), de g et de Y. En effet, l'absence d'isomorphisme calculable entre $\mathbb{G}_2$ et $\mathbb{G}_1$ empêche d'utiliser les autres éléments publics (à savoir $\widetilde{g}$, $\widetilde{X}$ et $\widetilde{Y}$) pour falsifier (h^*, t^*). Les éléments h^* et t^* ne pouvant être obtenus qu'au travers de requêtes à l'oracle d'opération de groupe de $\mathbb{G}_1$, nous connaissons $((u_{i,1}, v_{i,1}, u_{i,2}, v_{i,2})_i, (w_1, w_2), (w_1', w_2')) \in \mathbb{Z}_p^{4q+4}$ tels que :

$$g^{r^*} = h^* = g^{w_1} \cdot Y^{w_1'} \cdot \prod_{i=1}^{q} h_i^{u_{i,1}} \cdot t_i^{v_{i,1}} \quad \text{et} \quad g^{z^*} = t^* = g^{w_2} \cdot Y^{w_2'} \cdot \prod_{i=1}^{q} h_i^{u_{i,2}} \cdot t_i^{v_{i,2}},$$

ce qui entraîne que

$$r^* = w_1 + w_1' \cdot y + \sum_{i=1}^{q} (u_{i,1} \cdot r_i + v_{i,1}(x + y \cdot m_i) \cdot r_i)$$

$$\text{et} \quad z^* = w_2 + w_2' \cdot y + \sum_{i=1}^{q} (u_{i,2} \cdot r_i + v_{i,2}(x + y \cdot m_i) \cdot r_i).$$

La validité de cette nouvelle paire implique que $z^* = r^*(x + y \cdot m^*)$, et donc que :

$$\begin{aligned} w_2 + w_2' \cdot y + \sum_{i=1}^{q} (u_{i,2} \cdot r_i + v_{i,2}(x r_i + m_i \cdot y r_i)) \\ = w_1 \cdot x + w_1' \cdot xy + \sum_{i=1}^{q} (u_{i,1} \cdot x r_i + v_{i,1}(x^2 r_i + m_i \cdot xy r_i)) \\ + m^* \cdot (w_1 \cdot y + w_1' \cdot y^2 + \sum_{i=1}^{q} (u_{i,1} \cdot y r_i + v_{i,1}(xy r_i + m_i \cdot y^2 r_i))). \end{aligned}$$

Pour que ces deux polynômes soient égaux, il faut que leurs coefficients le soient. Ainsi :
- $v_{i,1} = 0\ \forall i$ car le membre de gauche ne comporte aucun monôme de degré 3 ;
- $u_{i,2} = 0\ \forall i$ car le membre de droite ne contient pas de terme en r_i ;
- $w_2 = 0$ car le membre de droite ne comporte pas de terme constant ;
- $w_1 = 0$ et $w'_1 = 0$ car le membre de gauche ne contient aucun terme en x ou xy.

L'égalité devient alors :

$$w'_2 \cdot y + \sum_{i=1}^{q} v_{i,2}(xr_i + m_i \cdot yr_i) = \sum_{i=1}^{q} u_{i,1} \cdot xr_i + m^* \cdot \sum_{i=1}^{q} u_{i,1} \cdot yr_i.$$

L'absence de terme en y dans le membre de droite implique alors que $w'_2 = 0$ et donc que :

$$\sum_{i=1}^{q} v_{i,2}(xr_i + m_i \cdot yr_i) = \sum_{i=1}^{q} u_{i,1} \cdot xr_i + m^* \cdot \sum_{i=1}^{q} u_{i,1} \cdot yr_i.$$

L'égalité des coefficients des monômes xr_i et yr_i implique que $v_{i,2} = u_{i,1}$ et $u_{i,1} \cdot m_i = u_{i,1} \cdot m^*$. Puisque $r^* \neq 0$ (autrement $h^* = 1_{\mathbb{G}_1}$), il existe au moins un scalaire $u_{i,1} = v_{i,2} \neq 0$ et donc $m^* = m_i$. La paire (h^*, t^*) n'est donc pas valide car elle porte sur un scalaire m^* déjà soumis à l'oracle. L'adversaire ne peut donc pas produire de nouvelle paire en combinant les éléments de $\mathbb{G}_1$ qu'il connait.

Il reste maintenant à évaluer la probabilité d'une égalité « accidentelle », c'est-à-dire le cas où deux polynômes distincts atteindraient la même image pour certaines valeurs de $x, y, r_1, \ldots, r_q \in \mathbb{Z}_p$. Les éléments retournés par $\mathcal{O}$ sont associés à des polynômes de degré, au plus, 2 et les éléments publics sont tous de degré inférieur ou égal à 1. Par conséquent toute combinaison obtenue par le biais des différentes opérations de groupe est de degré, au plus, 3 (en raison du couplage). Le nombre total de polynômes étant majoré par $5+2q+q_G$, il y a, au plus, $(5+2q+q_G)^2/2$ paires de polynômes distincts. D'après le lemme de Schwartz-Zippel, la probabilité que l'une de ces paires atteigne la même image est majorée par $3(5+2q+q_G)^2/2p$, ce qui est négligeable.

Le théorème 48 assure donc que le schéma permettant de signer un unique message est EUF-CMA sûr dans le modèle des groupes génériques. Le théorème suivant montre qu'il en va de même pour le schéma permettant de signer plusieurs messages.

Théorème 49. *Le schéma permettant de signer plusieurs messages est EUF-CMA sûr sous l'hypothèse 1. Plus précisément, si un adversaire peut en casser la sécurité avec probabilité ϵ alors il existe un adversaire, avec le même temps d'exécution et le même nombre de requêtes de signatures, pouvant casser la sécurité du schéma pour un unique message avec probabilité supérieure à $\epsilon - q/p$*

Preuve. Soit $\mathcal{A}$ un adversaire contre la sécurité EUF-CMA du protocole pour signer plusieurs messages. Nous construisons une réduction $\mathcal{R}$ utilisant $\mathcal{A}$ contre la sécurité EUF-CMA du protocole pour un unique message. Le challenger de ce dernier jeu sera noté $\mathcal{C}$.

- **Initialisation :** $\mathcal{R}$ reçoit de $\mathcal{C}$ une clé publique pk^* qui contient les paramètres publics *p.p.* ainsi que $(\widetilde{g}, \widetilde{X}, \widetilde{Y})$. Il sélectionne alors $\alpha_j, \beta_j \xleftarrow{\$} \mathbb{Z}_p$, pour $j = 1, \ldots, r$ et définit $\widetilde{Y}_j \leftarrow \widetilde{Y}^{\alpha_j} \widetilde{g}^{\beta_j}$. Pour finir, il envoie la clé publique $\mathsf{pk} \leftarrow (\widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r)$ du protocole pour signer plusieurs messages à $\mathcal{A}$.
- **Requêtes :** Lorsque $\mathcal{A}$ soumet une requête de signatures sur un vecteur $M_i = (m_{i,1}, \ldots, m_{i,r})$, $\mathcal{R}$ commence par demander à $\mathcal{C}$ une signature sur $m_i = \sum \alpha_j m_{i,j}$ et reçoit alors $\sigma = (\sigma_1, \sigma_2)$ tel que $e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{\sum \alpha_j m_{i,j}}) = e(\sigma_2, \widetilde{g})$. Il calcule ensuite $\sigma'_2 \leftarrow \sigma_2 \cdot \sigma_1^{\sum \beta_j \cdot m_{i,j}}$ et

retourne (σ_1, σ_2') à $\mathcal{A}$:

$$\begin{aligned} e(\sigma_2', \widetilde{g}) &= e(\sigma_2 \cdot \sigma_1^{\sum \beta_j \cdot m_{i,j}}, \widetilde{g}) = e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1^{\sum \beta_j \cdot m_{i,j}}, \widetilde{g}) \\ &= e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{\sum \alpha_j m_{i,j}}) \cdot e(\sigma_1, \widetilde{g}^{\sum \beta_j \cdot m_{i,j}}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}^{\alpha_j m_{i,j}} \widetilde{g}^{\beta_j \cdot m_{i,j}}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod (\widetilde{Y}^{\alpha_j} \widetilde{g}^{\beta_j})^{m_{i,j}}) = e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_{i,j}}), \end{aligned}$$

qui est donc bien une signature valide sur M_i.

– **Résultat :** Finalement, $\mathcal{A}$ retourne une signature $\sigma = (\sigma_1, \sigma_2)$ sur un vecteur $M^* = (m_1^*, \ldots, m_r^*)$. Celle ci est une falsification valide si :
  – $e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j^*}) = e(\sigma_2, \widetilde{g})$;
  – Pour $i = 1, \ldots, q$, $M^* \neq M_i$.

  Si $\sum \alpha_j m_j^* = \sum \alpha_j m_{i,j}$, pour un certain $i \in \{1, \ldots, q\}$, alors $\mathcal{R}$ arrête. Sinon, il retourne $\sigma^* = (\sigma_1^*, \sigma_2^*)$, avec $\sigma_1^* \leftarrow \sigma_1$ et $\sigma_2^* \leftarrow \sigma_2 \cdot \sigma_1^{-\sum \beta_j \cdot m_j^*}$, ainsi que $m^* \leftarrow \sum \alpha_j m_j^*$:

$$\begin{aligned} e(\sigma_2^*, \widetilde{g}) &= e(\sigma_2 \cdot \sigma_1^{-\sum \beta_j \cdot m_j^*}, \widetilde{g}) = e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1^{-\sum \beta_j \cdot m_j^*}, \widetilde{g}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j^*}) \cdot e(\sigma_1, \widetilde{g}^{-\sum \beta_j \cdot m_j^*}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod (\widetilde{Y}^{\alpha_j} \widetilde{g}^{\beta_j})^{m_j^*}) \cdot e(\sigma_1, \prod \widetilde{g}^{-\beta_j \cdot m_j^*}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}^{\alpha_j m_j^*}) = e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{\sum \alpha_j m_j^*}) = e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{m^*}). \end{aligned}$$

Puisque m^* n'a jamais été l'objet d'une requête de signature, il s'agit d'une falsification valide pour la clé publique pk^*.

Ainsi, $\mathcal{R}$ réussit à casser la sécurité EUF-CMA du schéma pour un unique message à moins que $\mathcal{A}$ ne retourne $M^* = (m_1^*, \ldots, m_r^*)$ tel que $\sum \alpha_j m_j^* = \sum \alpha_j m_{i,j}$ pour un certain vecteur $M_i = (m_{i,1}, \ldots, m_{i,r})$. Il reste donc à prouver qu'une telle relation est improbable.

Soit y le scalaire tel que $\widetilde{Y} = \widetilde{g}^y$, et soient $\gamma_j \xleftarrow{\$} \mathbb{Z}_p$, pour $j = 1, \ldots, r$. Si nous fixons $\alpha_j' \leftarrow \alpha_j - \gamma_j$ et $\beta_j' \leftarrow \beta_j + y\gamma_j$, for $j = 1, \ldots, r$, on remarque que :

$$\widetilde{Y}^{\alpha_j'} \widetilde{g}^{\beta_j'} = \widetilde{Y}^{\alpha_j - \gamma_j} \widetilde{g}^{\beta_j + y\gamma_j} = \widetilde{Y}^{\alpha_j} \widetilde{Y}^{-\gamma_j} \widetilde{g}^{\beta_j} \widetilde{Y}^{\gamma_j} = \widetilde{Y}^{\alpha_j} \widetilde{g}^{\beta_j} = \widetilde{Y}_j.$$

Ainsi, la clé publique pk est indépendante des scalaires γ_j, et ne révèle donc aucune information sur α_j. Il en va de même pour les signatures qui ne dépendent que de la clé publique et de l'élément σ_1 choisi par l'oracle.

Par conséquent, l'adversaire n'ayant accès qu'à des éléments indépendants de α_j, la probabilité que $\mathcal{R}$ arrête avant la fin du jeu est majorée par q/p.

8.2 Agrégation de Signatures

Une propriété intéressante de notre construction est qu'il est possible d'en dériver un schéma de signatures séquentiellement agrégeables. Cette primitive, introduite dans [LMRS04], est un cas particulier des signatures agrégéables proposées par Boneh *et al* [BGLS03].

Un schéma de signature agrégeable permet de fusionner plusieurs signatures sous des clés publiques différentes en une unique signature. Il peut se révéler intéressant dans le cas des infrastructures à clés publiques (PKI) où les utilisateurs reçoivent des *chaines de certificats* $(\sigma_1, m, \mathsf{pk}_1), (\sigma_2, \mathsf{pk}_1, \mathsf{pk}_2), \ldots, (\sigma_r, \mathsf{pk}_{r-1}, \mathsf{pk}_r)$, c'est-à-dire une liste de certificats où la clé publique de chaque autorité est elle-même certifiée par une autre autorité. Pour s'assurer de sa validité, l'utilisateur doit alors vérifier que σ_1 est bien une signature valide sur m sous la clé

publique pk_1, elle-même certifiée par une autorité dont la clé publique est pk_2 et ainsi de suite jusqu'à tomber sur une autorité en laquelle l'utilisateur a confiance. L'intérêt d'avoir recours à une signature agrégeable est donc double : cela permet de raccourcir la chaîne en envoyant une unique signature σ au lieu de $\sigma_1, \ldots, \sigma_r$ et de pouvoir vérifier en une seule fois que tous les messages (m ou pk_i) ont été signés.

Cependant, il peut être utile en pratique de recourir à une version plus restrictive, mais permettant des constructions plus efficaces, de ces signatures. C'est le cas des signatures séquentiellement agrégeables, où il n'est désormais plus possible de regrouper des signatures générées de manière indépendante mais seulement de fusionner une nouvelle signature à un agrégat déjà existant. Plus précisément, étant donnés m^*, sk^* et un agrégat σ (on parlera également de signature agrégée) prouvant qu'un ensemble de messages $m_1, \ldots, m_r$ a bien été signé sous des clés publiques respectives $\mathsf{pk}_1, \ldots, \mathsf{pk}_r$, il est possible de calculer un nouvel agrégat σ' prouvant que l'ensemble $m_1, \ldots, m_r, m^*$ a bien été signé sous $\mathsf{pk}_1, \ldots, \mathsf{pk}_r, \mathsf{pk}^*$.

Dans cette section, nous commençons par définir les signatures séquentiellement agrégeables et en présenter un modèle de sécurité. Nous décrivons ensuite comment rendre nos signatures séquentiellement agrégeables et prouvons la sécurité de la construction résultante.

8.2.1 Syntaxe et Modèle de Sécurité

Un schéma de signature séquentiellement agrégeable est défini par les algorithmes suivants :
- l'algorithme `AS.Setup` qui prend en entrée un paramètre de sécurité k et génère les paramètres publics *p.p.* du système ;
- l'algorithme `AS.Keygen` qui, prenant en entrée *p.p.*, génère une paire de clés $(\mathsf{sk}, \mathsf{pk})$ (on suppose que *p.p.* $\in \mathsf{pk}$ et que $\mathsf{pk} \in \mathsf{sk}$) ;
- l'algorithme `AS.Sign` qui, prenant en entrée une clé de signature sk, un message m et un agrégat σ sur des messages $(m_1, \ldots, m_r)$ pour des clés publiques $(\mathsf{pk}_1, \ldots, \mathsf{pk}_r)$ retourne un nouvel agrégat σ' sur $(m_1, \ldots, m_r, m)$;
- l'algorithme `AS.Verify` qui, prenant en entrée $(m_1, \ldots, m_r)$, σ et des clés publiques $(\mathsf{pk}_1, \ldots, \mathsf{pk}_r)$, retourne 1 si σ est une signature agrégée valide sur $(m_1, \ldots, m_r)$ pour ces clés publiques et 0 sinon.

Modèle de Sécurité

Comme les signatures numériques, la propriété de sécurité attendue d'un schéma de signature séquentiellement agrégeable est appelée *résistance aux falsifications existentielles contre des attaques adaptatives à messages choisis* et notée EUF-CMA. Elle est cependant définie différemment par le jeu suivant entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$.

- **Initialisation :** $\mathcal{C}$ commence par définir une liste `KeyList` qui contiendra les futures clés publiques puis exécute les algorithmes `AS.Setup` et `AS.Keygen` obtenant ainsi les paramètres publics *p.p.* et une paire de clés $(\mathsf{sk}^*, \mathsf{pk}^*)$. La clé publique pk^* est alors envoyée à $\mathcal{A}$.
- **Requêtes d'adhésion :** $\mathcal{A}$ envoie des clés publiques pk qui sont alors rajoutées à `KeyList`.
- **Requêtes de signatures :** $\mathcal{A}$ soumet, au plus, q requêtes de signatures sur des messages $m_1, \ldots, m_q$ sous la clé publique pk^*. A chaque fois, il fournit un agrégat σ_i portant sur des messages $(m_{i,1}, \ldots, m_{i,r_i})$ pour des clés publiques $(pk_{i,1}, \ldots, pk_{i,r_i})$ figurant toutes dans `KeyList`. $\mathcal{C}$ retourne alors le résultat de l'exécution de $\texttt{AS.Sign}(\mathsf{sk}^*, \sigma_i, (m_{i,1}, \ldots, m_{i,r_i}), (pk_{i,1}, \ldots, pk_{i,r_i}), m_i)$.
- **Résultat :** $\mathcal{A}$ produit finalement une signature agrégée σ sur les messages $(m_1^*, \ldots, m_r^*)$ pour les clés $(\mathsf{pk}_1, \ldots, \mathsf{pk}_r)$ et gagne le jeu si toutes les conditions suivantes sont vérifiées :
  - $\texttt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), (m_1^*, \ldots, m_r^*), \sigma) = 1$;
  - pour tout $\mathsf{pk}_j \neq \mathsf{pk}^*$, $\mathsf{pk}_j \in$ `KeyList` ;

– $\exists j^* \in [1, r]$ tel que $\mathsf{pk}^* = \mathsf{pk}_{j^*}$ et $m^*_{j^*}$ n'a pas fait l'objet d'une requête de signatures, *i.e.* $m^*_{j^*} \neq m_i$ pour tout $i = 1, \ldots, q$.

Un schéma de signature séquentiellement agrégeable est EUF-CMA sûr si la probabilité de succès de tout adversaire probabiliste polynomial est négligeable.

Remarque 25. Nous utiliserons dans cette section la variante de ce modèle proposée par Lu *et al.* [LOS+06] où l'adversaire doit prouver la connaissance de sk lors des requêtes d'adhésion. Cela correspond en pratique à une autorité de certification qui s'assurerait de la validité des clés publiques qu'elle enregistre.

8.2.2 Une Nouvelle Construction

Reprenons le schéma de signature, décrit dans la section 8.1.1, permettant de signer r messages. La clé secrète sk était composée de $r + 1$ scalaires $(x, y_1, \ldots, y_r)$. Supposons maintenant qu'une signature $(g, X) = (g, g^x)$, pour un certain $g \in \mathbb{G}_1$, sur le vecteur $(0, \ldots, 0)$ soit publiée. La connaissance de celle-ci ne fournit aucune information supplémentaire à un adversaire qui aurait pu, de toute façon, la demander dans le jeu de sécurité mais permet de supprimer x de la clé secrète. En effet, pour signer des messages $(m_1, \ldots, m_r)$, il suffit de générer un scalaire t aléatoire et de calculer $(g^t, (X)^t \cdot (g^t)^{\sum y_j \cdot m_j})$.

Il devient alors possible d'utiliser la technique de *partage de clé publique* proposée dans [LLY13] pour construire un schéma de signature séquentiellement agrégeable efficace. Le principe est que chaque signataire j génère sa propre paire de clés $(y_j, \widetilde{Y}_j) \in \mathbb{Z}_p \in \mathbb{G}_2$ mais utilise le même élément X désormais fourni dans les paramètres publics. Pour signer un message $m_1 \in \mathbb{Z}_p^*$, le premier signataire génère un scalaire t_1 aléatoire et retourne $(\sigma_1, \sigma_2) \leftarrow (g^{t_1}, (X)^{t_1} \cdot (g^{t_1})^{y_1 \cdot m_1})$. Pour y agréger une signature m_2, un deuxième signataire génère à son tour $t_2 \xleftarrow{\$} \mathbb{Z}_p$ puis calcule $(\sigma'_1, \sigma'_2) \leftarrow (\sigma_1^{t_2}, (\sigma_2 \cdot \sigma_1^{y_2 \cdot m_2})^{t_2})$. La signature agrégée ainsi obtenue peut alors s'écrire $(g^t, (g^t)^{x + m_1 \cdot y_1 + m_2 \cdot y_2})$ pour $t = t_1 \cdot t_2$ et sa validité peut donc être vérifiée à l'aide de l'algorithme `Verify` décrit dans la section 8.1.1.

Plus formellement, notre schéma de signature séquentiellement agrégeable est défini par les algorithmes suivants.

– $\mathtt{AS.Setup}(1^k)$: soit k le paramètre de sécurité, cet algorithme génère un scalaire aléatoire $x \in \mathbb{Z}_p$ et retourne $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, X, \widetilde{g}, \widetilde{X})$, où $X = g^x$ et $\widetilde{X} = \widetilde{g}^x$ pour des générateurs $(g, \widetilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$.

– $\mathtt{AS.Keygen}(p.p.)$: cet algorithme génère un $y \xleftarrow{\$} \mathbb{Z}_p$ aléatoire, calcule $\widetilde{Y} \leftarrow \widetilde{g}^y$ et définit $\mathsf{sk} = y$ et $\mathsf{pk} = \widetilde{Y}$.

– $\mathtt{AS.Sign}(\mathsf{sk}, \sigma, (m_1, \ldots, m_r), (\mathsf{pk}_1, \ldots, \mathsf{pk}_r), m)$: cet algorithme s'exécute de la manière suivante :
  – si $r = 0$, alors $\sigma \leftarrow (g, X)$;
  – si $r > 0$ mais $\mathtt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), \sigma, (m_1, \ldots, m_r)) = 0$, alors il s'arrête ;
  – si $m = 0$, il s'arrête ;
  – si $\exists j \in \{1, \ldots, r\}$ tel que $\mathsf{pk}_j = \mathsf{pk}$, alors il s'arrête.

  Si l'algorithme ne s'est pas arrêté, alors il définit $y = \mathsf{sk}$ et $(\sigma_1, \sigma_2) = \sigma$, génère $t \xleftarrow{\$} \mathbb{Z}_p$ et calcule $\sigma' = (\sigma'_1, \sigma'_2) \leftarrow (\sigma_1^t, (\sigma_2 \cdot \sigma_1^{y \cdot m})^t)$. Finalement, il retourne σ'.

– $\mathtt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), (m_1, \ldots, m_r), \sigma)$: cet algorithme définit $(\sigma_1, \sigma_2) = \sigma$ et $\widetilde{Y}_j = \mathsf{pk}_j$, pour $j = 1, \ldots, r$, et vérifie si les conditions $\sigma_1 \neq 1_{\mathbb{G}_1}$ et $e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j}) = e(\sigma_2, \widetilde{g})$ sont satisfaites. Il retourne 1 si c'est le cas et 0 sinon.

Les signatures agrégées ainsi construites ont donc la même forme que les signatures numériques décrites dans la section 8.1.1. Nous montrons d'ailleurs dans la prochaine section que la sécurité des signatures agrégées se ramène à celle de ces dernières.

Nous comparons dans le tableau ci-dessous l'efficacité de notre construction avec celle du schéma proposé dans [LLY13] qui utilise également la technique de *partage de clé publique*, mais sur les signatures CL. Notre schéma propose donc les signatures séquentiellement agrégeables les plus performantes, tant au niveau de la taille que du nombre d'opérations à effectuer. Il illustre par ailleurs toute la flexibilité de nos signatures qui se révèlent compatibles avec une technique initialement conçue pour les signatures CL.

	Taille	Coût (Sign.)	Coût (Vérif.)	Régén.	Groupes	Hypothèse
LLY [LLY13]	$3\,\mathbb{G}_1$	$1\,\mathtt{Ver.} + 5\,\mathtt{E}_{\mathbb{G}_1}$	$5\,\mathtt{P} + r\,\mathtt{E}_{\mathbb{G}_2}$	Oui	Tous	LRSW
Nous	$2\,\mathbb{G}_1$	$1\mathtt{Ver.} + 3\,\mathtt{E}_{\mathbb{G}_1}$	$2\,\mathtt{P} + r\,\mathtt{E}_{\mathbb{G}_2}$	Oui	Type 3	hypothèse 1

Table 8.2 – Comparaison entre notre schéma de signature agrégeable et celui de [LLY13]. Ici, r correspond au nombre de messages signés, $\mathtt{E}_{\mathbb{G}_i}$ au coût d'une exponentiation dans $\mathbb{G}_i$ ($i \in \{1, 2\}$), `Ver` au coût de vérification d'une signature agrégée et `Pair` au coût d'un calcul de couplage.

8.2.3 Étude de la Sécurité

Même si nos signatures agrégeables présentent la même forme que nos signatures numériques, la preuve de sécurité nécessite quelques précisions que nous donnons dans cette section.

Théorème 50. *Le schéma de signature séquentiellement agrégeable est EUF-CMA sûr sous l'hypothèse 1. Plus précisément, tout adversaire capable de casser la sécurité EUF-CMA de ce schéma peut être transformé en un adversaire réussissant à casser, avec la même probabilité, la sécurité EUF-CMA du schéma de signature numérique pour un unique message.*

Preuve. Soit $\mathcal{A}$ un adversaire contre la sécurité EUF-CMA du schéma de signature séquentiellement agrégeable. Nous construisons une réduction $\mathcal{R}$ utilisant $\mathcal{A}$ contre la sécurité EUF-CMA du schéma de signature pour un unique message. Le challenger de ce dernier jeu sera noté $\mathcal{C}$.

- **Initialisation :** $\mathcal{R}$ commence par créer une liste `KeyList` puis obtient de $\mathcal{C}$ une clé publique pk contenant les paramètres publics du schéma de signature ainsi qu'un triplet $(\widetilde{g}, \widetilde{X}, \widetilde{Y})$. Il soumet alors une requête de signature sur 0 à $\mathcal{C}$ qui lui envoie une paire (τ_1, τ_2) telle que $e(\tau_1, \widetilde{X}) = e(\tau_2, \widetilde{g})$. $\mathcal{R}$ définit alors $g \leftarrow \tau_1$ et $X \leftarrow \tau_2$ et envoie les paramètres publics $(g, X, \widetilde{g}, \widetilde{X})$ du schéma de signature agrégeable ainsi que $\mathsf{pk}^* \leftarrow \widetilde{Y}$ à $\mathcal{A}$. Dans ce qui suit, x^* et y^* désigneront les scalaires tels que $\widetilde{X} = g^{x^*}$ et $\widetilde{Y} = g^{y^*}$.
- **Requête d'adhésion :** Lorsque $\mathcal{A}$ souhaite rajouter une clé publique pk_i à la liste `KeyList`, il doit prouver la connaissance de la clé secrète sk_i correspondante, ce qui permet à $\mathcal{R}$ de l'extraire. Ce dernier connait donc toutes les clés secrètes associées aux éléments de la liste `KeyList`.
- **Requête de signature :** Lorsque $\mathcal{A}$ soumet une requête de signature sur un message m_i, il envoie un agrégeat σ_i portant sur des messages $(m_{i,1}, \ldots, m_{i,r_i})$ pour les clés publiques $(\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,r_i})$. Si $r_i > 0$, $\mathcal{R}$ commence par vérifier que σ_i est valide et s'arrête si ce n'est pas le cas. Il demande alors une signature sur m_i à $\mathcal{C}$ qui lui renvoie (σ_1, σ_2). Comme toutes les clés publiques $\mathsf{pk}_{i,j}$ ont été certifiées, $\mathcal{R}$ connait les clés secrètes $(\mathsf{sk}_{i,1}, \ldots, \mathsf{sk}_{i,r_i}) = (y_{i,1}, \ldots, y_{i,r_i})$ associées. Il sélectionne alors $t \xleftarrow{\$} \mathbb{Z}_p$ et retourne à $\mathcal{A}$:

$$\sigma' \leftarrow (\sigma_1^t, (\sigma_2 \cdot \sigma_1^{\sum_{j=1}^{r_i} y_{i,j} \cdot m_{i,j}})^t).$$

Cette signature est bien valide sur le vecteur $(m_{i,1}, \ldots m_{i,r_i}, m_i)$ pour les clés publiques

$(\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,r_i}, \mathsf{pk}^*)$ car :

$$\begin{aligned} e(\sigma'_2, \widetilde{g}) &= e((\sigma_2 \cdot \sigma_1^{\sum_{j=1}^{r_i} y_{i,j} \cdot m_{i,j}})^t, \widetilde{g}) = e(\sigma_2^t, \widetilde{g}) \cdot e(\sigma_1^t, \widetilde{g})^{\sum_{j=1}^{r_i} y_{i,j} \cdot m_{i,j}} \\ &= e(\sigma_2, \widetilde{g})^t \cdot \prod_{j=1}^{r_i} e(\sigma_1^t, \widetilde{g}^{y_{i,j} \cdot m_{i,j}}) = e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{m_i})^t \cdot \prod_{j=1}^{r_i} e(\sigma_1^t, \mathsf{pk}_{i,j}^{m_{i,j}}) \\ &= e(\sigma_1^t, \widetilde{X} \cdot \mathsf{pk}^{m_i} \cdot \prod_{j=1}^{r_i} \mathsf{pk}_{i,j}^{m_{i,j}}) = e(\sigma'_1, \widetilde{X} \cdot \prod_{j=1}^{r_i} \mathsf{pk}_{i,j}^{m_{i,j}} \cdot \mathsf{pk}^{m_i}). \end{aligned}$$

– **Résultat :** $\mathcal{A}$ finit par retourner une signature agrégée $\sigma = (\sigma_1, \sigma_2)$ sur des messages $(m_1^*, \ldots, m_r^*)$ pour les clés publiques $(\mathsf{pk}_1, \ldots, \mathsf{pk}_r)$. La signature σ est considérée comme une falsification valide si toutes les conditions suivantes sont satisfaites :

1. $\mathtt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), \sigma, (m_1^*, \ldots, m_r^*)) = 1$;
2. $\forall \mathsf{pk}_j \neq \mathsf{pk}^*$, $\mathsf{pk}_j \in \mathtt{KeyList}$;
3. $\exists j^* \in [1, r]$, $\mathsf{pk}^* = \mathsf{pk}_{j^*}$ et $m_{j^*}^*$ n'a fait l'objet d'aucune requête de signature, *i.e.* $m_{j^*}^* \neq m_i$, pour $i = 1, \ldots, q$.

La première condition implique que $e(\sigma_1, \widetilde{X} \prod \mathsf{pk}_j^{m_j^*}) = e(\sigma_2, \widetilde{g})$, tandis que la seconde signifie que $\mathcal{R}$ connaît y_j tel que $\mathsf{pk}_j = \widetilde{g}^{y_j}$, pour tout $j = 1, \ldots, r$, lorsque $\mathsf{pk}_j \neq \mathsf{pk}^*$. La dernière condition implique qu'il existe (un unique, puisque les clés publiques sont distinctes) $j^* \in [1, r]$ tel que $\mathsf{pk}^* = \mathsf{pk}_{j^*}$: $\mathcal{R}$ peut alors calculer $\sigma^* = (\sigma_1^* \leftarrow \sigma_1, \sigma_2^* \leftarrow \sigma_2 \cdot \prod_{j \neq j^*} \sigma_1^{-y_j \cdot m_j^*})$ qui vérifie :

$$\begin{aligned} e(\sigma_2^*, \widetilde{g}) &= e(\sigma_2 \cdot \prod_{j \neq j^*} \sigma_1^{-y_j \cdot m_j^*}, \widetilde{g}) = e(\sigma_2, \widetilde{g}) \cdot \prod_{j \neq j^*} e(\sigma_1^{-y_j \cdot m_j^*}, \widetilde{g}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \mathsf{pk}_j^{m_j^*}) \cdot \prod_{j \neq j^*} e(\sigma_1, \widetilde{g}^{-y_j \cdot m_j^*}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \mathsf{pk}_j^{m_j^*}) \cdot \prod_{j \neq j^*} e(\sigma_1, \widetilde{Y}_j^{-m_j^*}) \\ &= e(\sigma_1, \widetilde{X} \cdot \mathsf{pk}_{j^*}^{m_{j^*}^*}) = e(\sigma_1^*, \widetilde{X} \cdot \mathsf{pk}_{j^*}^{m_{j^*}^*}). \end{aligned}$$

Puisque $\mathsf{pk}^* = \mathsf{pk}_{j^*}$ et $m^* = m_{j^*}$ n'a pas été soumis, l'équation ci-dessus implique que σ^* est une falsification valide sur m^* pour la clé pk^*.

Il ne reste alors qu'à prouver que toutes les requêtes de signatures ont été correctement simulées. Une signature valide sur $(m_{i,1}, \ldots, m_{i,r_i}, m_i)$ pour des clés publiques $(\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,r_i}, \mathsf{pk}^*)$ est de la forme (σ_1, σ_2) où σ_1 est un élément aléatoire de $\mathbb{G}_1$ et σ_2 est l'unique élément de $\mathbb{G}_1$ tel que $e(\sigma_2, \widetilde{g}) = e(\sigma_1, \widetilde{X} \prod_{j=1}^{r_i} \widetilde{Y}_{i,j}^{m_{i,j}} \cdot (\mathsf{pk}^*)^{m_i})$. Par conséquent, la paire $\sigma' = (\sigma'_1, \sigma'_2)$ retournée à $\mathcal{A}$ est indistinguable d'une signature valide car σ'_1 a été généré à l'aide d'un scalaire t aléatoire et σ'_2 vérifie bien cette égalité. La simulation est donc parfaite.

8.3 Autres Applications

Nous avons vu dans la section précédente un exemple détaillé d'application de nos signatures. Il existe en réalité de nombreuses autres constructions, par exemple de signatures de groupe ou d'accréditations anonymes, qui pourraient bénéficier de celles-ci. Cependant, pour être utilisables, nos signatures doivent offrir d'autres fonctionnalités, telles que la possibilité de signer des messages mis en gage ou de prouver la connaissance de signatures. Nous décrivons dans cette section les protocoles correspondants et en illustrons l'utilisation à l'aide de deux exemples.

8.3.1 Signature de Messages Mis en Gage

Reprenons l'exemple du schéma de signature de groupe décrit dans [BCN+10]. Chaque utilisateur soumet, lors de son adhésion au groupe, une mise en gage $g^s \in \mathbb{G}_1$ de son secret $s \in \mathbb{Z}_p$ pour le faire certifier par le manager du groupe. Il reçoit alors une signature CL qui lui permettra de s'authentifier comme un utilisateur légitime.

Notre schéma de signature peut facilement gérer cette situation. L'utilisateur envoie g^s ainsi qu'une preuve de connaissance de s. Si cette dernière est valide, le manager du groupe peut alors retourner une signature $\sigma = (\sigma_1, \sigma_2) \leftarrow (g^u, (g^x \cdot (g^m)^y)^u))$ sur s. Comme nous le verrons un peu plus tard, il est en effet possible d'instancier la construction de [BCN+10] avec nos signatures, ce qui permet d'obtenir les signatures de groupe les plus courtes de l'état de l'art.

Cependant, pour certaines applications, révéler g^s peut être dangereux. C'est par exemple le cas des accréditations anonymes où s peut être un attribut appartenant à un petit ensemble qui pourrait être retrouvé par recherche exhaustive. Il est donc nécessaire de pouvoir signer des messages parfaitement masqués (au sens de la théorie de l'information) comme c'est le cas lorsqu'ils sont mis en gage en utilisant le protocole dû à Pedersen [Ped92]. Cela n'est cependant pas directement possible avec les protocoles décrits dans la section 8.1.1 car ils ne fournissent aucun élément du groupe $\mathbb{G}_1$ dans leur clé publique. Nous décrivons donc une nouvelle version adaptée à ce cas d'usage et présentons son extension permettant de signer plusieurs messages à la fois.

Protocole pour un unique message

Le protocole pour signer un unique message parfaitement masqué est constitué des algorithmes suivants :

- `Setup`(1^k) : étant donné un paramètre de sécurité k, cet algorithme retourne $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, la description de groupes bilinéaires de type 3.
- `Keygen`$(p.p.)$: cet algorithme sélectionne deux générateurs $g \xleftarrow{\$} \mathbb{G}_1$ et $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2$ ainsi que deux scalaires $(x, y) \xleftarrow{\$} \mathbb{Z}_p^2$. Il calcule ensuite $(X, Y) \leftarrow (g^x, g^y)$ et $(\widetilde{X}, \widetilde{Y}) \leftarrow (\widetilde{g}^x, \widetilde{g}^y)$, et définit $\mathsf{sk} \leftarrow X$ et $\mathsf{pk} \leftarrow (g, Y, \widetilde{g}, \widetilde{X}, \widetilde{Y})$.
- `Protocole` : un utilisateur souhaitant obtenir un message $m \in \mathbb{Z}_p$ choisit d'abord un scalaire $t \xleftarrow{\$} \mathbb{Z}_p$ aléatoire et calcule $C \leftarrow g^t Y^m$ qu'il envoie au signataire. Il démarre alors une preuve de connaissance de l'ouverture de cette mise en gage avec ce dernier qui, s'il est convaincu, choisit un scalaire $u \xleftarrow{\$} \mathbb{Z}_p$ et retourne $\sigma' \leftarrow (g^u, (XC)^u)$. L'utilisateur peut ensuite démasquer la signature en calculant $\sigma \leftarrow (\sigma'_1, \sigma'_2 / {\sigma'_1}^t)$.

L'élément σ satisfait alors $\sigma_1 = g^u$ et $\sigma_2 = (XC)^u / g^{ut} = (X g^t Y^m / g^t)^u = (XY^m)^u$, qui est une signature valide sur m pour le premier schéma décrit dans la section 8.1.1. En raison de la présence de nouveaux éléments dans la clé publique, la sécurité EUF-CMA de ce protocole reposera maintenant sur l'hypothèse 2.

Protocole pour plusieurs messages

Le protocole pour signer plusieurs messages parfaitement masqués est constitué des algorithmes suivants.

- `Setup`(1^k) : étant donné un paramètre de sécurité k, cet algorithme retourne $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, la description de groupes bilinéaires de type 3.
- `Keygen`$(p.p.)$: cet algorithme sélectionne deux générateurs $g \xleftarrow{\$} \mathbb{G}_1$ et $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2$ ainsi que des scalaires $(x, y_1, \ldots, y_r) \xleftarrow{\$} \mathbb{Z}_p^{r+1}$ avant de calculer $(X, Y_1, \ldots, Y_r) \leftarrow (g^x, g^{y_1}, \ldots, g^{y_r})$ et $(\widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r) \leftarrow (\widetilde{g}^x, \widetilde{g}^{y_1}, \ldots, \widetilde{g}^{y_r})$. Il définit alors $\mathsf{pk} \leftarrow (g, Y_1, \ldots, Y_r, \widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r)$ et $\mathsf{sk} \leftarrow X$.

– `Protocole` : un utilisateur souhaitant obtenir une signature sur les messages $(m_1, \ldots, m_r)$ sélectionne un scalaire aléatoire $t \xleftarrow{\$} \mathbb{Z}_p$ et calcule $C \leftarrow g^t \prod_{i=1}^{r} Y_i^{m_i}$ qu'il envoie au signataire. Il démarre alors une preuve de connaissance de l'ouverture de cette mise en gage avec ce dernier qui, s'il est convaincu, choisit un scalaire $u \xleftarrow{\$} \mathbb{Z}_p$ et retourne $\sigma' \leftarrow (g^u, (XC)^u)$. L'utilisateur peut alors démasquer la signature en calculant $\sigma \leftarrow (\sigma'_1, \sigma'_2/{\sigma'_1}^t)$.

De même, l'élément σ satisfait $\sigma_1 = g^u$ et $\sigma_2 = (XC)^u/g^{ut}$. En développant, on obtient $\sigma_2 = (Xg^t \prod_{i=1}^{r} Y_i^{m_i}/g^t)^u = (X \prod_{i=1}^{r} Y_i^{m_i})^u$, qui est donc une signature valide $(m_1, \ldots, m_r)$ pour le deuxième schéma décrit dans la section 8.1.1. Une preuve similaire à celle du théorème 49 permet de réduire la sécurité de ce protocole à celle du protocole précédent et donc à l'hypothèse 2.

Remarque 26. Il est important de remarquer que, pour les deux protocoles ci-dessus, la procédure de « démasquage » est optionnelle et ne doit être effectuée que pour les applications qui ne nécessitent pas de signatures indépendantes des messages signés. En effet, dans le premier protocole (cela reste vrai pour le second) l'élément σ' retourné par le signataire est en fait une signature valide sur le vecteur (m, t) sous la clé publique $(\widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \widetilde{Y}_2)$ avec $\widetilde{Y}_2 = \widetilde{g}$. Par conséquent σ' peut être lui-même utilisé chaque fois qu'une signature indépendante de m est nécessaire.

Inversement, il est possible de rendre n'importe quelle signature $\sigma \leftarrow (\sigma_1, \sigma_2)$ indépendante du message m signé en calculant $\sigma' \leftarrow (\sigma_1, \sigma_2 \cdot \sigma_1^t)$ pour un certain t aléatoire. Cela correspond en fait à agréger à σ une signature sur t pour un certain utilisateur dont la clé publique est $\widetilde{g}$.

8.3.2 Preuve de Connaissance d'une Signature

L'idée décrite dans la remarque précédente peut en fait être détournée pour construire une preuve de connaissance d'une signature. En effet, pour prouver connaissance d'une signature $\sigma = (\sigma_1, \sigma_2)$ sur $(m_1, \ldots, m_r)$, le prouveur $\mathcal{P}$:

1. choisit des scalaires aléatoires $r, t \xleftarrow{\$} \mathbb{Z}_p$ et calcule $\sigma' \leftarrow (\sigma_1^r, (\sigma_2 \cdot \sigma_1^t)^r)$;
2. envoie $\sigma' = (\sigma'_1, \sigma'_2)$ au vérificateur $\mathcal{V}$ avec qui il démarre une preuve de connaissance π de $(m_1, \ldots, m_r)$ et de t tels que :

$$e(\sigma'_1, \widetilde{X}) \cdot \prod e(\sigma'_1, \widetilde{Y}_j)^{m_j} \cdot e(\sigma'_1, \widetilde{g})^t = e(\sigma'_2, \widetilde{g})$$

Le vérificateur accepte alors si π est valide.

La preuve π étant une preuve de connaissance de logarithmes discrets dans le groupe $\mathbb{G}_T$ (qui est d'ordre premier), il est possible d'utiliser une extension du protocole de Schnorr décrit dans la section 3.

En agrégeant à σ une signature sur un « message » t aléatoire, le prouveur fait de σ'_2 un élément aléatoire de $\mathbb{G}_1$. De même la régénération de la signature, obtenue par l'élévation de chacun de ses éléments à la puissance r, permet de rendre σ'_1 aléatoire. Cela va permettre à $\mathcal{P}$ de simuler facilement chacun des éléments impliqués dans la preuve. Plus formellement, nous obtenons le résultat suivant.

Théorème 51. *Le protocole décrit ci-dessus est une preuve de connaissance à divulgation nulle d'une signature σ sur des messages $(m_1, \ldots, m_r)$.*

Preuve. La complétude de la preuve provient directement de celle de π ainsi que de la remarque précédente.

La propriété de divulgation nulle provient de l'existence du simulateur $\mathcal{S}$ qui commence par générer deux éléments aléatoires σ'_1 et σ'_2 de $\mathbb{G}_1$ et définit $\sigma' = (\sigma'_1, \sigma'_2)$ avant d'exécuter le simulateur de la preuve π. La simulation est parfaite car r et t sont générés aléatoirement dans le

protocole, ce qui fait de σ'_1 et σ'_2 des éléments aléatoires de $\mathbb{G}_1$, et car π est elle-même une preuve à divulgation nulle.

Finalement, considérons un prouveur P capable de convaincre $\mathcal{V}$ avec une probabilité non négligeable. Nous construisons alors un extracteur $\mathcal{E}$ utilisant $\mathcal{P}$ pour obtenir une signature valide σ sur un vecteur de messages. Comme π est une preuve de connaissance, $\mathcal{E}$ peut exécuter l'extracteur associé pour retrouver un vecteur $(m_1, \dots m_r)$ ainsi que t vérifiant la relation :

$$e(\sigma'_1, \widetilde{X}) \prod e(\sigma'_1, \widetilde{Y}_j)^{m_j} \cdot e(\sigma'_1, \widetilde{g})^t = e(\sigma'_2, \widetilde{g}).$$

$\mathcal{E}$ peut alors calculer $\sigma = (\sigma_1, \sigma_2) \leftarrow (\sigma'_1, \sigma'_2 \cdot (\sigma'_1)^{-t})$ qui est une signature valide sur le vecteur $(m_1, \dots, m_r)$ puisque :

$$\begin{aligned} e(\sigma_1, \widetilde{X}) \cdot \prod e(\sigma_1, \widetilde{Y}_j)^{m_j} &= e(\sigma'_1, \widetilde{X}) \cdot \prod e(\sigma'_1, \widetilde{Y}_j)^{m_j} \cdot e(\sigma'_1, \widetilde{g})^t \cdot e(\sigma'_1, \widetilde{g})^{-t} \\ &= e(\sigma'_2, \widetilde{g}) \cdot e(\sigma'_1, \widetilde{g})^{-t} \\ &= e(\sigma_2, \widetilde{g}). \end{aligned}$$

L'existence d'un extracteur valide implique donc que le protocole décrit ci-dessus est bien une preuve de connaissance.

8.3.3 Exemples

Nous décrivons dans cette section deux exemples d'utilisation de nos signatures dans des protocoles plus complexes. Nous commençons par une application aux signatures de groupes avant de nous intéresser aux accréditations anonymes.

Signature de Groupe

Le schéma proposé par Bischel *et al* [BCN+10] offre les signatures de groupe les plus courtes de l'état de l'art. Son efficacité est notamment due à l'utilisation des signatures CL et à leur capacité à être régénérées. Nous montrons dans le tableau 8.3 qu'il est possible de faire encore mieux en instanciant leur schéma avec nos signatures comme nous le décrivons ci-dessous. On peut noter que le schéma original nécessitait déjà l'utilisation de groupes bilinéaires de type 3, le recours à nos signatures n'entraîne donc aucune nouvelle contrainte.

- $\texttt{GSetup}(1^k)$: Le manager du groupe exécute les algorithmes `Setup` et `Keygen` du schéma de signature pour un unique message décrit dans la section 8.1.1 obtenant ainsi $\mathsf{sk} = (x, y)$ et $\mathsf{pk} = (\widetilde{g}, \widetilde{X}, \widetilde{Y})$. Il définit alors la clé publique du groupe gpk comme étant pk ainsi qu'un certain générateur $g \in \mathbb{G}_1$ et sa clé secrète gmsk comme étant sk.
- $\texttt{PKIJoin}(i, 1^k)$: L'utilisateur i génère une paire de clés $(\mathsf{usk}[i], \mathsf{upk}[i]) \leftarrow \Sigma.\texttt{Keygen}(1^k)$ pour un certain schéma de signature numérique Σ et envoie $\mathsf{upk}[i]$ à une autorité de certification. On suppose désormais que $\mathsf{upk}[i]$ est publiquement consultable et que tout le monde peut en obtenir une copie authentique.
- `GJoin` : Lorsqu'il rejoint le groupe, l'utilisateur i démarre un protocole interactif avec le manager du groupe. Il commence par générer un secret $\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p$ et envoie la paire $(\tau, \widetilde{\tau}) \leftarrow (g^{\mathsf{sk}_i}, \widetilde{Y}^{\mathsf{sk}_i})$ ainsi qu'une signature $\eta \leftarrow \Sigma.\texttt{Sign}(\mathsf{usk}[i], \tau)$ au manager du groupe. Ce dernier en vérifie alors la validité ainsi que celle de la paire $(\tau, \widetilde{\tau})$ en testant si $e(\tau, \widetilde{Y}) = e(g, \widetilde{\tau})$. L'utilisateur démarre ensuite une preuve interactive de connaissance de sk_i, telle que celle de Schnorr (décrit dans la figure 3.2 de la section 3). Si le manager du groupe est convaincu, il génère alors un scalaire u aléatoire et calcule une signature $\sigma = (\sigma_1, \sigma_2) \leftarrow (g^u, (g^x \cdot (\tau)^y)^u)$ sur sk_i comme expliqué au début de la section 8.3.1. Finalement, le manager de groupe conserve $(i, \tau, \eta, \widetilde{\tau})$ dans un registre secret et envoie σ à l'utilisateur qui définit sa clé secrète gsk_i comme étant $(\mathsf{sk}_i, \sigma, e(\sigma_1, \widetilde{Y}))$. On peut noter cependant que ce dernier

n'a pas réellement besoin de conserver $e(\sigma_1, \widetilde{Y})$ mais cela lui évitera d'avoir à calculer des couplages lorsqu'il exécutera l'algorithme `GSign`.

- `GSign`(gsk_i, m) : Pour signer un message m l'utilisateur commence par régénérer la signature σ en choisissant un scalaire t aléatoire et en calculant $(\sigma'_1, \sigma'_2) \leftarrow (\sigma_1^t, \sigma_2^t)$. Il doit alors produire une signature de connaissance de sk_i. Pour cela, il génère un $k \xleftarrow{\$} \mathbb{Z}_p$ aléatoire et calcule $e(\sigma'_1, \widetilde{Y})^k \leftarrow e(\sigma_1, \widetilde{Y})^{k \cdot t}$ et $c \leftarrow \mathcal{H}(\sigma'_1, \sigma'_2, e(\sigma_1, \widetilde{Y})^{k \cdot t}, m)$ pour une certaine fonction de hachage $\mathcal{H}$ qui sera modélisée par un oracle aléatoire dans la preuve de sécurité. Il calcule ensuite $s \leftarrow k + c \cdot \mathsf{sk}_i \bmod p$ et retourne finalement la signature de groupe $\mu = (\sigma'_1, \sigma'_2, c, s) \in \mathbb{G}_1^2 \times \mathbb{Z}_p^2$ sur m.
- `GVerify`(gpk_i, m) : Pour vérifier une signature de groupe $\mu = (\sigma_1, \sigma_2, c, s)$ sur m, on commence par calculer $R \leftarrow (e(\sigma_1^{-1}, \widetilde{X}) \cdot e(\sigma_2, \widetilde{g}))^{-c} \cdot e(\sigma_1^s, \widetilde{Y})$ et par tester si $c = \mathcal{H}(\sigma_1, \sigma_2, R, m)$ ce qui correspond à la vérification de la signature de connaissance. On retourne 1 si le test est valide et 0 sinon. La complétude de ce protocole provient du fait que, si (σ_1, σ_2) est une signature valide sur sk_i, alors :

$$\begin{aligned}
&(e(\sigma_1^{-1}, \widetilde{X}) \cdot e(\sigma_2, \widetilde{g}))^{-c} \cdot e(\sigma_1^s, \widetilde{Y}) \\
&\qquad = e(\sigma_1, \widetilde{Y})^k \cdot [e(\sigma_1, \widetilde{Y})^{\mathsf{sk}_i} \cdot e(\sigma_1, \widetilde{X}) \cdot e(\sigma_2^{-1}, \widetilde{g})]^c \\
&\qquad = e(\sigma_1, \widetilde{Y})^k \cdot [e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{\mathsf{sk}_i}) \cdot e(\sigma_2, \widetilde{g})^{-1}]^c \\
&\qquad = e(\sigma_1, \widetilde{Y})^k
\end{aligned}$$

- `GOpen`(gmsk, m, μ) : Pour ouvrir une signature μ, le manager du groupe teste, pour toutes les entrées $(i, \tau_i, \eta_i, \widetilde{\tau}_i)$ de son registre, si $e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1, \widetilde{X})^{-1} = e(\sigma_1, \widetilde{\tau})$ jusqu'à ce que l'égalité soit vérifiée. Il retourne alors le triplet (i, τ_i, η_i) correspondant ainsi qu'une preuve de connaissance de $\widetilde{\tau}_i$ qui pourra être vérifiée par toute personne souhaitant s'assurer de la validité de l'ouverture.

Signature de Groupe	Taille	Coût (Sign.)	Coût (Vérif.)
Bichsel *et al* [BCN+10]	3 $\mathbb{G}_1$ + 2 $\mathbb{Z}_p$	3 $\mathtt{E}_{\mathbb{G}_1}$ + 1 $\mathtt{E}_{\mathbb{G}_T}$ + 1 `H`	5 `P` + 1 $\mathtt{E}_{\mathbb{G}_1}$ + 1 $\mathtt{E}_{\mathbb{G}_T}$
Nous	2 $\mathbb{G}_1$ + 2 $\mathbb{Z}_p$	2 $\mathtt{E}_{\mathbb{G}_1}$ + 1 $\mathtt{E}_{\mathbb{G}_T}$ + 1 `H`	3 `P` + 1 $\mathtt{E}_{\mathbb{G}_1}$ + 1 $\mathtt{E}_{\mathbb{G}_T}$

Table 8.3 – Comparaison entre le schéma de signature de groupe décrit dans [BCN+10] et celui instancié avec nos signatures. La notation $\mathtt{E}_{\mathbb{G}_1}$ (resp. $\mathtt{E}_{\mathbb{G}_T}$) fait référence au coût d'une exponentiation dans $\mathbb{G}_1$ (resp. $\mathbb{G}_T$), `P` au coût d'un calcul de couplage et `H` au coût d'évaluation d'une fonction de hachage dans $\mathbb{Z}_p$. Nous ne tenons pas compte du coût des opérations dans $\mathbb{Z}_p$ qui est négligeable par rapport à celui des autres.

Accréditations Anonymes

Un système d'accréditations anonymes permet à un utilisateur de prouver que certains de ses attributs (par exemple, son âge) sont certifiés sans rien révéler d'autre à son sujet. Dans l'idéal, les utilisations répétées d'une même accréditation ne devraient pas pouvoir être tracées. Par ailleurs, il peut être souhaitable de pouvoir faire certifier ses attributs ainsi que de prouver certaines affirmations les concernant (par exemple âge ≥ 18) sans les révéler. Dans un environnement bilinéaire, le schéma proposé dans [CL04] et l'extension de [BBS04] décrite dans [ASM06] satisfont chacune de ces conditions. Certains schémas (tel que celui de [HS14]) offrent une efficacité remarquable mais au prix d'au moins l'une de ces propriétés.

Comme expliqué dans [CL04], un protocole d'accréditations anonymes nécessite un schéma de signature capable de signer des messages mis en gage et pour lequel il est possible de prouver efficacement la connaissance d'une signature. Nos signatures proposant ces fonctionnalités,

elles peuvent être utilisées. On obtient alors un système d'accréditation anonyme se comparant favorablement aux autres systèmes de l'état de l'art, comme l'illustre le tableau 8.4.

Accrédit. Anonymes	Émission		Preuve		
	Utilisateur	Émetteur	Utilisateur	Verifieur	Données
CL [CL04]	$(r+1)\,\mathtt{E}_{\mathbb{G}_1} + PK\{\mathtt{E}_{\mathbb{G}_1}[r+1]\}$	$(2r+4)\,\mathtt{E}_{\mathbb{G}_1} + \mathtt{Ver}(PK)$	$(2r+4)\,\mathtt{E}_{\mathbb{G}_1} + PK\{\mathtt{P}[r+2]\}$	$(4r+2)\,\mathtt{P} + \mathtt{Ver}(PK)$	$(2r+3)\,\mathbb{G}_1 + \lvert PK\rvert$
BBS+ [ASM06]	$(r+1)\,\mathtt{E}_{\mathbb{G}_1} + PK\{\mathtt{E}_{\mathbb{G}_1}[r+1]\}$	$2\,\mathtt{E}_{\mathbb{G}_1} + \mathtt{Ver}(PK)$	$3\,\mathtt{E}_{\mathbb{G}_1} + PK\{\mathtt{P}[r+4] + \mathtt{E}_{\mathbb{G}_1}[3] + \mathtt{E}_{\mathbb{G}_1}[2]\}$	$\mathtt{Ver}(PK)$	$2\,\mathbb{G}_1 + \lvert PK\rvert$
Nous	$(r+1)\,\mathtt{E}_{\mathbb{G}_1} + PK\{\mathtt{E}_{\mathbb{G}_1}[r+1]\}$	$2\,\mathtt{E}_{\mathbb{G}_1} + \mathtt{Ver}(PK)$	$2\,\mathtt{E}_{\mathbb{G}_1} + PK\{\mathtt{P}[r+1]\}$	$\mathtt{Ver}(PK)$	$2\mathbb{G}_1 + \lvert PK\rvert$

Table 8.4 – Comparaison entre un système d'accréditations anonymes utilisant nos signatures et l'état de l'art. La notation r fait référence au nombre d'attributs à certifier, $\mathtt{E}_{\mathbb{G}_1}$ au coût d'une exponentiation dans $\mathbb{G}_1$, $\mathtt{P}$ au coût d'un calcul de couplage, $PK\{\mathtt{E}_{\mathbb{G}_1}[n]\}$ (resp. $PK\{\mathtt{P}[n]\}$) au coût d'une preuve de connaissance de n scalaires secrets impliqués dans une équation d'exponentiations multiples (resp. de produit de couplages), $\mathtt{Ver}(PK)$ au coût de vérification de cette preuve et $|PK|$ à la taille de ses traces.

Conclusion

Nous avons étudié dans ce mémoire différentes approches pour concilier sécurité, anonymat et efficacité dans nos usages quotidiens.

Nous avons tout d'abord considéré le cas du paiement pour lequel plusieurs primitives (toutes issues du concept de monnaie électronique introduit par Chaum) ont déjà été proposées. Parmi elles nous avons identifié la monnaie électronique divisible comme celle offrant les fonctionnalités les plus intéressantes. Malheureusement, les protocoles existant pour l'instancier souffraient soit d'un relâchement des exigences de sécurité, soit d'un problème de performances. Nous avons alors décrit deux nouvelles constructions anonymes et sûres, pouvant satisfaire des contraintes d'efficacité très strictes. En effet, une implémentation sur carte SIM de l'une d'entre elles a prouvé que les dépenses d'un montant inférieur à 100 € pouvait s'effectuer en moins de 500 ms. Il n'y a donc plus de réels obstacles techniques au déploiement d'une solution de paiement électronique anonyme.

Suivant la même idée, nous nous sommes intéressés à la problématique un peu plus générale de l'authentification anonyme. Au travers de l'exemple du transport public, nous avons cherché à illustrer toute la flexibilité de la cryptographie moderne, capable de répondre à des besoins extrêmement précis. La primitive que nous avons introduite dans le chapitre 5 montre notamment qu'il est possible de placer des garde-fous pour empêcher les différentes entités impliquées dans un système d'abuser de leurs pouvoirs.

Cependant, même les protocoles les plus efficaces de l'état de l'art peuvent se révéler trop complexes pour des périphériques de faible puissance. Il est alors nécessaire d'étudier d'autres pistes pour surmonter ce problème.

L'une d'entre elles consiste à alléger la charge des périphériques les moins performants en en transférant une partie vers des entités plus puissantes. Cette notion de délégation se révèle d'autant plus pertinente dans un monde où un nombre croissant d'objets se retrouvent connectés.

Cependant, les problèmes de sécurité soulevés par l'apparition de nouvelles entités dans un système sont complexes et nécessitent d'étudier chaque cas séparément. Pour permettre une réutilisation des résultats obtenus, il est donc préférable de s'intéresser à la délégation d'algorithmes fréquemment employés. C'est notamment le cas des preuves de connaissance, utilisés massivement dans les protocoles anonymes, pour lesquelles nous avons proposé un protocole coopératif.

Pour aller plus loin dans cette stratégie, il est possible de s'intéresser directement à la délégation des briques élémentaires d'un algorithme, à savoir les opérations mathématiques qui le composent. Cette approche se révèle d'autant plus intéressante que l'opération est coûteuse. Nous avons donc cherché à déléguer l'une des plus complexes utilisées en cryptographie, à savoir le couplage bilinéaire. Le schéma que nous avons proposé est capable de garantir toutes les propriétés de sécurité attendues d'un protocole coopératif tout en offrant, dans bon nombre de cas, un gain significatif d'efficacité par rapport à l'exécution standard de l'algorithme.

Malheureusement, il n'est pas toujours possible de déléguer, il faut donc étudier d'autres solutions pour optimiser les performances d'un protocole. Dans le dernier chapitre de ce mémoire, nous avons donc cherché à instancier plus efficacement une brique cryptographique majeure, la signature numérique. Nous avons montré qu'il était possible de concevoir une signature parti-

culièrement courte dotée de multiples fonctionnalités qui la rendent utilisable par une grande variété de protocoles. Ces derniers bénéficient alors d'un gain important d'efficacité facilitant leur implémentation sur des périphériques contraints.

Problèmes Ouverts. Les travaux que nous avons menés sur la monnaie électronique divisible montrent qu'il est encore possible, même pour une primitive vieille de plus de 20 ans, de proposer des nouvelles constructions sensiblement plus efficaces. Il est donc naturel de se demander s'il est possible de faire encore mieux, notamment en se débarrassant du coût logarithmique des dépenses. En effet, tous les schémas modélisent les pièces sous la forme d'arbres binaires imposant de décomposer chaque montant V comme une somme de puissance de 2. Le poids de Hamming de V détermine alors le nombre de fois que le protocole doit être répété. Trouver un moyen pour l'utilisateur de révéler une unique information permettant à la banque de reconstituer les V numéros de série (que V soit une puissance de 2 ou non), sans que cela nuise à l'anonymat, constituerait une véritable avancée.

Nous avons considéré dans ce mémoire que la monnaie électronique divisible était la piste la plus sérieuse pour déployer un système de monnaie électronique anonyme. L'argument était qu'elle semblait la plus à même de régler les problèmes d'appoints. Cependant un schéma de monnaie électronique transférable pourrait également être une solution s'il était efficace. Ce type de système souffre certes d'une taille de pièce croissant avec le nombre de transactions [CP93], mais la borne théorique est en fait très faible et parfaitement compatible avec une utilisation en pratique. Malheureusement, les constructions de l'état de l'art sont loin de l'atteindre, ce qui laisse à penser qu'il existe une forte marge de progression pour cette primitive.

Dans le domaine de la délégation d'opérations, il est frappant de voir qu'il n'existe toujours pas, malgré de très nombreux travaux de recherche, de solutions qui soient satisfaisantes à tous les points de vue. L'efficacité de notre protocole de délégation de couplages dépend par exemple beaucoup des paramètres des courbes elliptiques utilisées. Cependant, l'évolution des performances des différentes solutions est assez significative et laisse espérer de nouveaux gains dans les prochaines années.

Finalement, dans le but d'instancier plus efficacement les briques cryptographiques, une dernière question serait de savoir s'il est possible de faire mieux que les preuves Groth-Sahai pour construire des preuves non-interactives dans le modèle standard. Celles-ci sont en effet systématiquement utilisées dans les environnements bilinéaires et pourtant leur complexité est loin d'être négligeable. Certains travaux [JR13, LPJY14, JR14] montrent qu'il est possible d'être plus efficace dans le cas des preuves *quasi-adaptatives*, mais un résultat similaire pour le cas général reste encore à trouver.

Publications Personnelles

Nos publications dans des conférences internationales sont listées ci-dessous. Parmi elles, figure l'article *Forward Secure Non-Interactive Key Exchange* [PS14], publié lors de la conférence SCN 2014. Nous y avons étudié les échanges de clés non-interactifs dotés d'une propriété additionnelle, à savoir que chaque utilisateur dispose d'une clé secrète évoluant avec le temps malgré une clé publique constante. Le point important est que la connaissance d'une clé secrète à un instant t ne permet pas d'obtenir d'information sur les clés secrètes précédemment utilisées. Ce sujet ne correspond cependant pas aux thèmes développés dans ce mémoire, ce qui explique pourquoi il n'y a pas été abordé.

[PS15] *Short Randomizable Signatures.* (**Travail en cours**)
D. Pointcheval et O. Sanders.

[CPST15b] *Scalable Divisible E-Cash.* (**ACNS 2015**)
S. Canard, D. Pointcheval, O. Sanders et J. Traoré.

[CPST15a] *Divisible E-Cash Made Practical.* (**PKC 2015**)
S. Canard, D. Pointcheval, O. Sanders et J. Traoré.

[DLST14] *Direct Anonymous Attestation with Dependent Basename Opening.* (**CANS 2014**)
N. Desmoulins, R. Lescuyer, O. Sanders et J. Traoré.

[PS14] *Forward Secure Non-Interactive Key Exchange.* (**SCN 2014**)
D. Pointcheval et O. Sanders.

[CDS14] *Delegating a Pairing can be Both Secure and Efficient.* (**ACNS 2014**)
S. Canard, J. Devigne et O. Sanders.

[CPS14] *Efficient Delegation of Zero-Knowledge Proofs of Knowledge in a Pairing-Friendly Setting.* (**PKC 2014**)
S. Canard, D. Pointcheval et O. Sanders.

[CCD+13] *Toward Generic Method for Server-Aided Cryptography.* (**ICICS 2013**)
S. Canard, I. Coisel, J. Devigne, C. Gallais, T. Peters et O. Sanders.

[AS12] *Efficient Group Signatures in the Standard Model.* (**ICISC 2012**)
L. El Aimani et O. Sanders.

Brevets

Procédé de conversion d'un premier chiffré en un deuxième chiffré.
Brevet déposé le 22/06/2015 avec S. Canard.

Procédé d'Encapsulation d'une Clé de Session.
Brevet déposé le 04/12/2014 avec S. Canard.

Procédé de Délégation de Couplages Bilinéaires.
Brevet déposé le 11/06/2013 avec S. Canard et J. Devigne.

Procédés et Dispositifs de Signatures de Groupe Cryptographiques.
Brevet déposé le 18/09/12 avec L. El Aimani.

Liste des tableaux

Liste des figures

Bibliographie

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer, August 2000.

[Adl79] Leonard M. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography (abstract). In *20th Annual Symposium on Foundations of Computer Science*, pages 55–60. IEEE Computer Society, 1979.

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, April / May 2002.

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236. Springer, August 2010.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 649–666. Springer, August 2011.

[ANS13] Référentiel général de sécurité. Technical report, Agence Nationale de la Sécurité des Systèmes d'Information, 2013.

[AS12] Laila El Aimani and Olivier Sanders. Efficient group signatures in the standard model. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *Information Security and Cryptology - ICISC 2012*, volume 7839 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2012.

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06 : 5th International Conference on Security in Communication Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, September 2006.

[ASM08] Man Ho Au, Willy Susilo, and Yi Mu. Practical anonymous divisible e-cash from bounded accumulators. In Gene Tsudik, editor, *FC 2008 : 12th International Conference on Financial Cryptography and Data Security*, volume 5143 of *Lecture Notes in Computer Science*, pages 287–301. Springer, January 2008.

[AW04] Michel Abdalla and Bogdan Warinschi. On the minimal assumptions of group signature schemes. In Javier López, Sihan Qing, and Eiji Okamoto, editors, *ICICS 04 : 6th International Conference on Information and Communication Security*, volume 3269 of *Lecture Notes in Computer Science*, pages 1–13. Springer, October 2004.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*,

volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, May 2004.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2) :149–177, April 2008.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, August 2004.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04 : 11th Conference on Computer and Communications Security*, pages 132–145. ACM Press, October 2004.

[BCLY08] Vicente Benjumea, Seung Geol Choi, Javier Lopez, and Moti Yung. Fair traceable multi-group signatures. In Gene Tsudik, editor, *FC 2008 : 12th International Conference on Financial Cryptography and Data Security*, volume 5143 of *Lecture Notes in Computer Science*, pages 231–246. Springer, January 2008.

[BCN+10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10 : 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398. Springer, September 2010.

[BCN14] Joppe W. Bos, Craig Costello, and Michael Naehrig. Exponentiating in pairing groups. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013 : 20th Annual International Workshop on Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Computer Science*, pages 438–455. Springer, August 2014.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, August 2001.

[BFG+13] David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. *Int. J. Inf. Sec.*, 12(3) :219–249, 2013.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, August 1993.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, May 2003.

[BGM+10] Jean-Luc Beuchat, Jorge E. González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-speed software implementation of the optimal Ate pairing over Barreto-Naehrig curves. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *PAIRING 2010 : 4th International Conference on Pairing-based Cryptography*, volume 6487 of *Lecture Notes in Computer Science*, pages 21–39. Springer, December 2010.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005 : 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, February 2005.

[BGOS07] Paulo S. L. M. Barreto, Steven D. Galbraith, Colm O'Eigeartaigh, and Michael Scott. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography*, 42(3) :239–271, 2007.

[BK98] R. Balasubramanian and Neal Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the Menezes - Okamoto - Vanstone algorithm. *Journal of Cryptology*, 11(2) :141–145, 1998.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13 : 20th Conference on Computer and Communications Security*, pages 1087–1098. ACM Press, November 2013.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, December 2001.

[BLS03] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02 : 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 257–267. Springer, September 2003.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures : Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, May 2003.

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography, 12th International Workshop, SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005 : 12th Annual International Workshop on Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, August 2006.

[BPV98] Victor Boyko, Marcus Peinado, and Ramarathnam Venkatesan. Speeding up discrete log and factoring based schemes via precomputations. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 221–235. Springer, May / June 1998.

[BQ95] Philippe Béguin and Jean-Jacques Quisquater. Fast server-aided RSA signatures secure against active attacks. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 57–69. Springer, August 1995.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical : A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93 : 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.

[BS04] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04 : 11th Conference on Computer and Communications Security*, pages 168–177. ACM Press, October 2004.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures : The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, February 2005.

[BT99] Fabrice Boudot and Jacques Traoré. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In Vijay Varadharajan and Yi Mu, editors, *Information and Communication Security, Second International Conference, ICICS'99*, volume 1726 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 1999.

[Cam97] Jan Camenisch. Efficient and generalized group signatures. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 465–479. Springer, May 1997.

[CCD+13] Sébastien Canard, Iwen Coisel, Julien Devigne, Cécilia Gallais, Thomas Peters, and Olivier Sanders. Toward generic method for server-aided cryptography. In Sihan Qing, Jianying Zhou, and Dongmei Liu, editors, *Information and Communications Security - 15th International Conference, ICICS 2013*, volume 8233 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2013.

[CCdMP10] Sébastien Canard, Iwen Coisel, Giacomo de Meulenaer, and Olivier Pereira. Group signatures are suitable for constrained devices. In Kyung Hyune Rhee and DaeHun Nyang, editors, *Information Security and Cryptology - ICISC 2010*, volume 6829 of *Lecture Notes in Computer Science*, pages 133–150. Springer, 2010.

[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*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer, May 2013.

[CCM+05] Benoît Chevallier-Mames, Jean-Sébastien Coron, Noel McCullagh, David Naccache, and Michael Scott. Secure delegation of elliptic-curve pairing. *IACR Cryptology ePrint Archive*, 2005 :150, 2005.

[CCM+10] Benoît Chevallier-Mames, Jean-Sébastien Coron, Noel McCullagh, David Naccache, and Michael Scott. Secure delegation of elliptic-curve pairing. In Gollmann et al. [GLI10], pages 24–35.

[CDS14] Sébastien Canard, Julien Devigne, and Olivier Sanders. Delegating a pairing can be both secure and efficient. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14 : 12th International Conference on Applied Cryptography and Network Security*, volume 8479 of *Lecture Notes in Computer Science*, pages 549–565. Springer, June 2014.

[CFA+12] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2012.

[CFT98] Agnes Hui Chan, Yair Frankel, and Yiannis Tsiounis. Easy come - easy go divisible cash. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer, May / June 1998.

[CG07] Sébastien Canard and Aline Gouget. Divisible e-cash systems can be truly anonymous. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 482–497. Springer, May 2007.

[CG10] Sébastien Canard and Aline Gouget. Multiple denominations in e-cash with compact transaction data. In Radu Sion, editor, *FC 2010 : 14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 82–97. Springer, January 2010.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, May 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In Moni Naor, editor, *TCC 2004 : 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 40–57. Springer, February 2004.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer, May 2005.

[CHL+14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehle. Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906, 2014. http://eprint.iacr.org/.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, August 2004.

[CL13] Sébastien Canard and Roch Lescuyer. Protecting privacy by sanitizing personal data : a new approach to anonymous credentials. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS 13 : 8th Conference on Computer and Communications Security*, pages 381–392. ACM Press, May 2013.

[CLM+12] Xiaofeng Chen, Jin Li, Jianfeng Ma, Qiang Tang, and Wenjing Lou. New algorithms for secure outsourcing of modular exponentiations. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS 2012 : 17th European Symposium on Research in Computer Security*, volume 7459 of *Lecture Notes in Computer Science*, pages 541–556. Springer, September 2012.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493. Springer, August 2013.

[CLT14a] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/975, 2014. http://eprint.iacr.org/.

[CLT14b] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In Hugo Krawczyk, editor, *PKC 2014 : 17th International Workshop on Theory and Practice in Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 311–328. Springer, March 2014.

[CM11] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings - the role of Ψ revisited. *Discrete Applied Mathematics*, 159(13) :1311–1322, 2011.

[CM14] Sanjit Chatterjee and Alfred Menezes. Type 2 structure-preserving signature schemes revisited. *IACR Cryptology ePrint Archive*, 2014 :635, 2014.

[CP93] David Chaum and Torben P. Pedersen. Transferred cash grows in size. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 390–407. Springer, May 1993.

[CP95] Lidong Chen and Torben Pryds Pedersen. New group signature schemes (extended abstract). In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer, May 1995.

[CPP05] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 542–558. Springer, May 2005.

[CPS10] Liqun Chen, Dan Page, and Nigel P. Smart. On the design and implementation of an efficient DAA scheme. In Gollmann et al. [GLI10], pages 223–237.

[CPS14] Sébastien Canard, David Pointcheval, and Olivier Sanders. Efficient delegation of zero-knowledge proofs of knowledge in a pairing-friendly setting. In Hugo Krawczyk, editor, *PKC 2014 : 17th International Workshop on Theory and Practice in Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 167–184. Springer, March 2014.

[CPST15a] Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Divisible e-cash made practical. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 77–100. Springer, 2015.

[CPST15b] Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Scalable divisible e-cash. In *ACNS '15*, Lecture Notes in Computer Science. Springer, 2015. Full version available on Cryptology ePrint Archive, http://eprint.iacr.org/.

[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*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, August 1997.

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, April 1991.

[DES77] Data encryption standard. Technical report, US National Bureau of Standard, 1977.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976.

[DLST14] Nicolas Desmoulins, Roch Lescuyer, Olivier Sanders, and Jacques Traoré. Direct anonymous attestations with dependent basename opening. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *CANS 14 : 13th International Conference on Cryptology and Network Security*, volume 8813 of *Lecture Notes in Computer Science*, pages 206–221. Springer, October 2014.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progressin Cryptology - VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.

[dR91] Peter de Rooij. On the security of the schnorr scheme using preprocessing. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 71–80. Springer, 1991.

[dR97] Peter de Rooij. On schnorr's preprocessing for digital signature schemes. *J. Cryptology*, 10(1) :1–16, 1997.

[Duc10] Léo Ducas. Anonymity from asymmetry : New constructions for anonymous HIBE. In Josef Pieprzyk, editor, *Topics in Cryptology – CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 148–164. Springer, March 2010.

[ElG84] Taher ElGamal. 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*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, August 1984.

[FFS87] Uriel Feige, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 210–217. ACM Press, May 1987.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself : Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, August 1987.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, May / June 2009.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, May 2013.

[GHMS14] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes : Cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929, 2014. http://eprint.iacr.org/.

[GL05] Marc Girault and David Lefranc. Server-aided verification : Theory and practice. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 605–623. Springer, December 2005.

[GLI10] Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors. *Smart Card Research and Advanced Application, 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010*, volume 6035 of *Lecture Notes in Computer Science*. Springer, 2010.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2) :281–308, 1988.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1) :186–208, 1989.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16) :3113–3121, 2008.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, December 2007.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.

[GSM12] GSMA. White paper : Mobile nfc in transport. Technical report, GSMA, 2012.

[Gui13] Aurore Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13 : 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 357–372. Springer, June 2013.

[GV14] Aurore Guillevic and Damien Vergnaud. Algorithms for outsourcing pairing computation. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014*, volume 8968 of *Lecture Notes in Computer Science*, pages 193–211. Springer, 2014.

[HL05] Susan Hohenberger and Anna Lysyanskaya. How to securely outsource cryptographic computations. In Joe Kilian, editor, *TCC 2005 : 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 264–282. Springer, February 2005.

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 491–511. Springer, December 2014.

[HSV06] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10) :4595–4602, 2006.

[IL13] Malika Izabachène and Benoît Libert. Divisible E-cash in the standard model. In Michel Abdalla and Tanja Lange, editors, *PAIRING 2012 : 5th International Conference on Pairing-based Cryptography*, volume 7708 of *Lecture Notes in Computer Science*, pages 314–332. Springer, May 2013.

[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.

[JR13] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 1–20. Springer, December 2013.

[JR14] Charanjit S. Jutla and Arnab Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 295–312. Springer, August 2014.

[Kah96] David Kahn. *The codebreakers : the story of secret writing.* Scribner, New York, 1996.

[KLP05] Bo Gyeong Kang, Moon Sung Lee, and Je Hong Park. Efficient delegation of pairing computation. *IACR Cryptology ePrint Archive*, 2005 :259, 2005.

[KM10] Neal Koblitz and Alfred Menezes. The brave new world of bodacious assumptions in cryptography. *Notices of the American Mathematical Society*, 57(3) :357–365, 2010.

[KSS08] Ezekiel J. Kachisa, Edward F. Schaefer, and Michael Scott. Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008 : 2nd International Conference on Pairing-based Cryptography*, volume 5209 of *Lecture Notes in Computer Science*, pages 126–135. Springer, September 2008.

[KTY04] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer, May 2004.

[KY06] Aggelos Kiayias and Moti Yung. Secure scalable group signature with dynamic joins and separable authorities. *IJSN*, 1(1/2) :24–45, 2006.

[LJ14] Benoît Libert and Marc Joye. Group signatures with message-dependent opening in the standard model. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 286–306. Springer, February 2014.

[LLY13] Kwangsu Lee, Dong Hoon Lee, and Moti Yung. Aggregating CL-signatures revisited : Extended functionality and better efficiency. In Ahmad-Reza Sadeghi, editor, *FC 2013 : 17th International Conference on Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 171–188. Springer, April 2013.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90. Springer, May 2004.

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, May / June 2006.

[LPJY14] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Non-malleability from malleability : Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 514–532. Springer, May 2014.

[LRSW00] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999 : 6th Annual International Workshop on Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer, August 2000.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite : More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 239–256. Springer, May 2014.

[MB02] Greg Maitland and Colin Boyd. Co-operatively formed group signatures. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 218–235. Springer, February 2002.

[MKI90] Tsutomu Matsumoto, Koki Kato, and Hideki Imai. Speeding up secret computations with insecure auxiliary devices. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 497–506. Springer, August 1990.

[MVO91] Alfred Menezes, Scott A. Vanstone, and Tatsuaki Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *23rd Annual ACM Symposium on Theory of Computing*, pages 80–89. ACM Press, May 1991.

[NS98] Phong Q. Nguyen and Jacques Stern. The Béguin-Quisquater server-aided RSA protocol from Crypto '95 is not secure. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 372–379. Springer, October 1998.

[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.

[NS01] Phong Q. Nguyen and Igor Shparlinski. On the insecurity of a server-aided RSA protocol. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 21–35. Springer, December 2001.

[NSS00] Phong Q. Nguyen, Igor E. Shparlinski, and Jacques Stern. Distribution of modular sums and the security of the server aided exponentiation, 2000.

[Oka95] Tatsuaki Okamoto. An efficient divisible electronic cash scheme. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 438–451. Springer, August 1995.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 324–337. Springer, August 1992.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, August 1992.

[PS96] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 252–265. Springer, November 1996.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3) :361–396, 2000.

[PS14] David Pointcheval and Olivier Sanders. Forward secure non-interactive key exchange. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14 : 9th International Conference on Security in Communication Networks*, volume 8642 of *Lecture Notes in Computer Science*, pages 21–39. Springer, September 2014.

[PS15] David Pointcheval and Olivier Sanders. Short randomizable signatures. *IACR Cryptology ePrint Archive*, 2015 :525, 2015. http://eprint.iacr.org/.

[PW93] Birgit Pfitzmann and Michael Waidner. Attacks on protocols for server-aided RSA computation. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 153–162. Springer, May 1993.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2) :120–126, 1978.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, August 1990.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3) :161–174, 1991.

[Sco13] Michael Scott. Unbalancing pairing-based key exchange protocols. *IACR Cryptology ePrint Archive*, 2013 :688, 2013.

[SEH+13] Yusuke Sakai, Keita Emura, Goichiro Hanaoka, Yutaka Kawai, Takahiro Matsuda, and Kazumasa Omote. Group signatures with message-dependent opening. In Michel Abdalla and Tanja Lange, editors, *PAIRING 2012 : 5th International Conference on Pairing-based Cryptography*, volume 7708 of *Lecture Notes in Computer Science*, pages 270–294. Springer, May 2013.

[SHA02] Secure hash standard. publication fips 180-2. Technical report, National Institute of Standards and Technology, 2002.

[SHA14] Secure hash standard. publication fips 202. Technical report, National Institute of Standards and Technology, 2014.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, May 1997.

[SR13] Ana Helena Sánchez and Francisco Rodríguez-Henríquez. NEON implementation of an attribute-based encryption scheme. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13 : 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 322–338. Springer, June 2013.

[TW87] Martin Tompa and Heather Woll. Random self-reducibility and zero knowledge interactive proofs of possession of information. In *28th Annual Symposium on Foundations of Computer Science*, pages 472–482. IEEE Computer Society Press, October 1987.

[TZR15] Haibo Tian, Fangguo Zhang, and Kun Ren. Secure bilinear pairing outsourcing made more efficient and flexible. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, pages 417–426, New York, NY, USA, 2015. ACM.

[vDCG+06] Marten van Dijk, Dwaine E. Clarke, Blaise Gassend, G. Edward Suh, and Srinivas Devadas. Speeding up exponentiation using an untrusted computational resource. *Des. Codes Cryptography*, 39(2) :253–273, 2006.

[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*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, May 2010.

[Ver10] Frederik Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1) :455–461, 2010.

[WWW+14] Yujue Wang, Qianhong Wu, Duncan S. Wong, Bo Qin, Sherman S. M. Chow, Zhen Liu, and Xiao Tan. Securely outsourcing exponentiations with single untrusted program for cloud storage. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014 : 19th European Symposium on Research in Computer Security, Part I*, volume 8712 of *Lecture Notes in Computer Science*, pages 326–343. Springer, September 2014.

[ZDN10] ZDNET. Hong kong e-payment firm admits selling customer data. www.zdnet.com/article/hong-kong-e-payment-firm-admits-selling-customer-data/. 2010.

Résumé

Les nouvelles technologies ont profondément modifié nos usages mais ne sont pas que synonymes d'avantages pour leurs utilisateurs. Elles ont en effet de lourdes conséquences sur notre vie privée, ce qui est bien souvent sous-estimé. Les utilisateurs de moyens de paiement électronique ne réalisent par exemple pas toujours que leurs transactions peuvent révéler des informations particulièrement intimes à leur sujet, telles que leur localisation, leur état de santé ou mêmes leurs croyances.

Nous nous intéressons dans ce mémoire aux techniques cryptographiques permettant de concilier les exigences de sécurité traditionnelles et le respect de la vie privée. Dans une première partie nous étudions deux cas particuliers, celui du paiement anonyme et celui de l'authentification anonyme. Nous proposons de nouvelles constructions qui offrent une meilleure efficacité que les solutions existantes, ouvrant ainsi la voie à de réelles applications pratiques. Chacun de ces systèmes fait l'objet d'une étude de sécurité montrant qu'ils offrent de solides garanties sous des hypothèses raisonnables.

Cependant, afin de satisfaire des contraintes techniques souvent très fortes dans ces contextes, il peut être nécessaire d'optimiser ces constructions qui nécessitent souvent un nombre significatif de calculs. Dans une deuxième partie nous proposons donc des moyens pour améliorer l'efficacité des opérations et algorithmes les plus fréquemment utilisés. Chacune de ces contributions peut présenter un intérêt au-delà du contexte de l'anonymat.

Abstract

New technologies offer greater convenience for end-users but usually at the cost of a loss in terms of privacy, which is often underestimated by the latter. For example, knowledge by a third party of the information related to a transaction is far from insignificant since it may reveal intimate details such as whereabouts, religious beliefs or health status.

In this thesis, we are interested in cryptographic technics allowing to reconcile both security requirements and user's privacy. In a first part, we will focus on two specific cases : anonymous payment and anonymous authentication. We propose new constructions, improving the efficiency of state-of-the-art solutions, which make all the features of these primitives more accessible for practical applications. We provide a detailed security analysis for each scheme, proving that they achieve the expected properties under reasonable assumptions.

However, to fulfill the strong technical constraints of these use cases, it may be necessary to optimize these constructions which are usually rather complex. To this end, we propose in a second part, new solutions to improve the efficiency of most common operations and algorithms. Each of these contributions is not restricted to anonymous systems and thus may be of independent interest.