ANNÉE 2017

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MathSTIC

présentée par

Solenn Brunet

Préparée à l'unité de recherche CIDRe

Confidentialité, Intégrité, Disponibilité et Répartition
IRISA - UMR 6074

Conception de mécanismes d'accréditations anonymes et d'anonymisation de données

Thèse soutenue à Rennes
le 27 novembre 2017

devant le jury composé de :

Benjamin NGUYEN
Professeur, INSA Centre Val de Loire / *rapporteur*

Olivier PEREIRA
Professeur, Université Catholique de Louvain / *rapporteur*

Carlos AGUILAR MELCHOR
Maître de conférences, Université de Toulouse / *examinateur*

Pierre-Alain FOUQUE
Professeur, Université de Rennes 1 / *examinateur*

Cristina ONETE
Maître de conférences, Université de Limoges / *examinatrice*

Jacques TRAORÉ
Ingénieur de recherche, Orange Labs / *examinateur*

Christophe BIDAN
Professeur, CentraleSupélec / *directeur de thèse*

Sébastien GAMBS
Professeur, Université du Québec à Montréal / *co-directeur de thèse*

Conception de mécanismes d'accréditations anonymes et d'anonymisation de données

Solenn BRUNET

Travaux effectués au sein de l'Applied Crypto Group, Orange Labs,
et de l'Institut de Recherche en Informatique et Systèmes Aléatoires, Rennes

Table des matières

Notations

$\mathbb{Z}$	Ensemble des entiers relatifs
$\mathbb{Z}_n$	Ensemble des entiers entre 0 et $n-1$
$\{0,1\}^*$	Ensemble des chaînes de longueur arbitraire composées de 0 et de 1
$\{0,1\}^k$	Ensemble des chaînes de longueur k composées de 0 et de 1
$\perp$	Chaîne de caractères vide
$\lvert X \rvert$	Cardinal de l'ensemble X
$\wedge$	Opérateur logique "et"
$\vee$	Opérateur logique "ou"
$a \leftarrow b$	La valeur de b est affectée à a
$x \xleftarrow{R} X$	L'élément x est choisi aléatoirement dans l'ensemble X selon une distribution uniforme
$x \leftarrow \mathcal{A}(\dots)$	L'algorithme $\mathcal{A}$ produit l'élément x en sortie
$\texttt{Proto}(\mathcal{U}_1(in_1),\mathcal{U}_2(in_2))$	Protocole interactif `Proto` entre $\mathcal{U}_1$ et $\mathcal{U}_2$ avec in_1 et in_2 comme entrée respective
$\text{pgcd}(a,b)$	Plus grand commun diviseur des deux entiers non nuls a et b
$\text{ppcm}(a,b)$	Plus petit commun multiple des deux entiers non nuls a et b
$\mathbb{P}(S)$	Probabilité de l'évènement S
$\mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2}(in)$	Adversaire $\mathcal{A}$ avec l'entrée in, ayant accès aux oracles $\mathcal{O}_1$ et $\mathcal{O}_2$

Introduction

L'avènement de l'informatique puis celui de l'Internet ont entraîné de nouveaux usages, accompagnés de nouveaux défis en termes de sécurité et de vie privée. De nos jours, les nouvelles technologies sont adoptées et plébiscitées par le grand public pour les avantages qu'elles apportent. Ainsi, il est maintenant possible d'utiliser une carte bancaire ou même un téléphone mobile pour réaliser des paiements, sans avoir nécessairement des espèces avec soi. Une carte d'abonnement aux transports nous évite d'acheter des tickets à usage unique et de les stocker. De même, grâce au télépéage, il n'est plus nécessaire de s'arrêter à chaque borne. L'utilisation des courriers électroniques ou d'applications de messagerie nous permet de communiquer en temps réel avec l'étranger. Bien d'autres applications émergent régulièrement et la cryptographie joue un rôle clé pour leur sécurité.

La cryptographie est une science ancienne qui répond au besoin de protéger les communications et qui est apparue dès l'invention de l'écriture. Son but initial est alors d'assurer la confidentialité des messages échangés, essentiellement à des fins militaires. Le chiffrement est l'opération qui va permettre de rendre un texte inintelligible, sauf pour le destinataire choisi. Les premiers mécanismes reposent sur le fait que l'expéditeur et le destinataire possèdent une même clé pour chiffrer et déchiffrer les messages grâce à un mélange de permutations et de substitutions connus d'eux uniquement.

Dès la fin du XIXème siècle, KERCKHOFFS critique ce mode de fonctionnement qui repose sur le secret de l'algorithme de chiffrement. Il affirme que la sécurité d'un mécanisme cryptographique ne doit dépendre que de la clé utilisée alors que l'algorithme peut être connu de tous et garantissant un haut niveau de sécurité. Le développement des communications démontre ce besoin d'utiliser un même protocole entre plusieurs entités. Avoir des algorithmes publics, étudiés et éprouvés par la communauté scientifique, permet d'établir des standards cryptographiques à disposition de tous. Les premiers schémas publics considèrent alors que le secret est la clé, connue de l'expéditeur et du destinataire, pour chiffrer et déchiffrer. Ce type de cryptographie est dite symétrique ou à clé secrète, en référence à cette clé commune. Néanmoins, communiquer la clé entre les entités devient plus complexe dans le monde numérique où l'utilisation d'un support physique pour la transmettre n'est plus réaliste. En réponse à ce problème d'échange de clé, DIFFIE et HELLMAN [DH76] introduisent la cryptographie dite asymétrique ou à clé publique. Un même destinataire détient alors une clé publique, utilisable par tous afin de chiffrer un message lui étant destiné, et une clé privée, connue de lui seul, qui permet de déchiffrer ses messages. Dans ce cas, connaître la clé publique ne représente pas de danger et ne dévoile aucune information. Ces deux modes de chiffrement, à clé publique et à clé secrète, peuvent alors être employés conjointement pour bénéficier de la sécurité du chiffrement à clé publique et de la rapidité du chiffrement à clé secrète.

Dès lors, la cryptographie n'est plus limitée à un usage militaire. Elle doit également assurer l'intégrité et l'authenticité des messages dans un monde où les données sont échangées et circulent à grande vitesse. Ainsi, il est possible de vérifier que le message reçu n'a pas été modifié au cours de son transfert grâce aux codes d'authentification de messages et qu'il a bien été émis par l'expéditeur attendu à partir de sa signature numérique. La cryptographie est devenue aujourd'hui omniprésente et se retrouve dans notre téléphone, notre ordinateur ou encore nos cartes à puces. Les défis à relever sont nombreux, comme la non-répudiation ou l'anonymat, et ils évoluent constamment.

Cryptographie et protection de la vie privée

Les nouvelles technologies apportent des avantages évidents, avec des services innovants et même personnalisés. Cependant, ceux-ci peuvent également entraîner des risques importants en terme de respect de la vie privée lorsqu'ils impliquent une collecte de données à caractère personnel. Le paiement électronique ou l'abonnement de télépéage n'incluent pas aisément l'anonymat garanti par leurs équivalents physiques. Les utilisateurs n'ont pas toujours conscience des informations personnelles et sensibles qui peuvent être déduites de leurs traces laissées sur les réseaux, volontairement ou non. À partir de données de mobilité par exemple, partagées sur un réseau social ou communiquées à une application, il est possible d'inférer le domicile de l'individu, ses activités favorites, son cercle d'amis ou encore des informations plus spécifiques comme des problèmes de santé, ses opinions politiques ou encore ses croyances religieuses [ZB11, JPBN09, NS09]. Étudier des données de paiement est tout aussi révélateur.

Assurer l'authentification de l'utilisateur tout en garantissant son anonymat n'est pas trivial dans le monde numérique. Chaque application et chaque type de données a ses propres spécificités et doit faire l'objet d'une étude particulière. Outre l'anonymat, les propriétés à assurer ne sont pas les mêmes d'un cas d'usage à l'autre selon la finalité et le support utilisé. De plus, étant donné qu'il est question d'applications pratiques, l'efficacité des solutions proposées est essentielle. En effet, si une solution cryptographique permet d'atteindre l'ensemble des propriétés souhaitées mais que ses performances ne sont pas suffisantes, elle ne sera pas adoptée pour des déploiements dans des produits utilisés au quotidien.

Par ailleurs, il est parfois inévitable que des données soient collectées par nécessité de service ou par obligation légale. Elles représentent alors une grande richesse qu'il faut manipuler avec précaution. Les étudier apportent de nombreuses informations pour améliorer le quotidien ou anticiper au mieux différents évènements [PWWH16, SGAW17]. Cela représente également un gain pour le propriétaire qui peut alors exploiter et valoriser ses données. Par exemple, dans le cas de la mobilité, différents types d'études peuvent être réalisées : scientifiques, pour l'étude de la mobilité ou de l'évolution de certaines maladies ; sociétale, pour déterminer si un nouvel arrêt de bus ou un rond-point est nécessaire à un endroit donné ; ou encore économique, pour choisir où implanter une nouvelle boutique selon la clientèle visée. Cependant, ces analyses doivent être réalisées dans un cadre précis et requièrent des mécanismes d'anonymisation de données. L'objectif est alors d'assainir les données afin de préserver au maximum leur utilité mais en protégeant aussi la vie privée des utilisateurs. Si l'assainissement est correctement réalisé, il est difficile de trouver un lien entre les données et les individus concernés. Toutefois, le compromis entre respect de la vie privée et utilité des données finales

est difficile à satisfaire puisque de nombreuses informations sont à disposition sur les réseaux et peuvent mener à des inférences.

Nos Contributions

Le but de cette thèse est de montrer comment des mécanismes cryptographiques ou des techniques d'assainissement de données peuvent permettre de concilier à la fois le respect de la vie privée, les exigences de sécurité et l'utilité du service fourni. Le mémoire est organisé en trois parties dont la première est consacrée aux outils mathématiques et aux principales primitives cryptographiques sur lesquelles reposent nos contributions.

Ensuite, dans la deuxième partie, nous nous intéressons aux mécanismes permettant d'offrir des propriétés fortes de respect de la vie privée dans plusieurs cas d'usage.

Nouvelles primitives. Pour satisfaire au critère d'efficacité, essentiel en pratique, il est nécessaire de concevoir de nouvelles primitives cryptographiques adaptées aux périphériques que nous utilisons aujourd'hui comme la carte SIM de notre téléphone.

Dans le Chapitre 3, nous proposons trois nouvelles constructions efficaces et prouvées sûres. En 2014, CHASE, MEIKLEJOHN et ZAVERUCHA [CMZ14] introduisent les codes d'authentification de messages (MACs) avec une structure de groupe. Un MAC est une primitive cryptographique qui permet d'authentifier des messages où une seule clé sert à la fois à sa construction et à sa vérification. Il diffère donc de la signature numérique qui peut être vérifiée par tous. La particularité de leur schéma de MAC est qu'il peut être vu comme une signature numérique car des paramètres publics peuvent être générés pour la vérification. Cependant, leur schéma nécessite autant de clés que de messages. Nous proposons alors un nouveau MAC algébrique où une seule clé suffit quelque soit le nombre de messages authentifiés. Ensuite, nous introduisons un MAC dit "séquentiellement agrégé" qui offre la possibilité de combiner efficacement les MAC générés par plusieurs utilisateurs au sein d'une même valeur. Enfin, nous présentons une nouvelle signature "partiellement aveugle" qui ne présente pas de risques de traçage même si elle est utilisée plusieurs fois. Chacune de ces primitives peut alors être utilisée pour construire des applications respectueuses de la vie privée comme des accréditations anonymes.

Accréditations anonymes. Les systèmes d'accréditations anonymes, introduits par CHAUM [Cha82], sont des outils essentiels pour assurer la protection de la vie privée des utilisateurs vis à vis du fournisseur de services. En effet, un tel système permet à un utilisateur de prouver que ses attributs sont certifiés, sans révéler aucune information supplémentaire. Par exemple, un étudiant peut attester qu'il a le droit au tarif réduit à un guichet, sans pour autant révéler son établissement d'inscription ou sa date de naissance. Ces schémas reposent en général sur des signatures numériques.

Nous nous intéressons en particulier à leur variante dite "avec vérification par clé" où le vérificateur a connaissance de la clé secrète qui a permis de générer l'accréditation. À partir de notre MAC algébrique, nous présentons un nouveau schéma d'accréditations anonymes prouvé sûr, dans le Chapitre 4. Il est compatible avec les périphériques actuels et limite le nombre de clés requises. Son efficacité est alors comparable aux solutions les plus connues actuellement [Paq13, PZ13, IBM10] mais assure en plus que l'utilisation successive d'une même accréditation ne représente pas de risque de traçage. En outre, il peut être adapté simplement

en une version classique où n'importe quel vérificateur a la capacité de tester la validité de l'accréditation, sans conséquence négative pour l'utilisateur.

Ces résultats ont été publiés à la conférence SAC 2016, dans l'article intitulé *"Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials"* [BBDT16].

Vote électronique. L'une des applications classiques des accréditations anonymes est le vote électronique. Transposer le système de vote physique à une version numérique soulève de nombreux problèmes puisque le vote standard repose à la fois sur le bulletin papier, l'urne scellée et l'isoloir où chacun est totalement libre de son choix. Dans le monde numérique, il devient bien plus difficile de garantir à l'électeur la confidentialité de son vote tout en lui permettant de vérifier que celui-ci a bien été pris en compte pour le résultat du scrutin et qu'il n'a pas été modifié. De la même façon, tout le monde doit pouvoir s'assurer que le vote a lieu dans des conditions normales et que seuls les bulletins valides sont comptabilisés.

Dans le Chapitre 5, nous travaillons en particulier à atteindre la propriété de résistance à la coercition qui découle de l'absence d'isoloir. En effet, un adversaire pourrait chercher à forcer les votants à choisir un candidat en particulier ou à lui remettre une preuve numérique du vote réalisé. À partir de notre MAC séquentiellement agrégé, nous proposons un nouveau schéma de vote électronique, prouvé sûr, qui satisfait cette propriété. Chaque étape du scrutin est publiquement vérifiable et le votant est capable de vérifier que son vote a été comptabilisé. Il est aussi suffisamment efficace pour être utilisable dans de vraies élections.

Ces résultats ont été publiés au *workshop* VOTING'16, associé à la conférence FC 2016, dans l'article *"Remote Electronic Voting can be Efficient, Verifiable and Coercion-Resistant"* [ABBT16].

Paiement électronique. Une deuxième application classique des accréditations anonymes est le paiement électronique. De la même façon, celui-ci soulève des problèmes d'anonymat, de duplication du moyen de paiement, de gestion du rendu de monnaie ou encore de la capacité à transférer une somme entre deux participants. La monnaie électronique, introduite par CHAUM [Cha82], a pour objectif de reproduire notre monnaie physique ainsi que l'anonymat des transactions, entre un utilisateur, un marchand et une banque. Selon les propriétés recherchées, les systèmes de monnaie électronique ont été déclinés en plusieurs catégories et souffrent généralement de problème d'efficacité.

Nous nous intéressons à un type particulier de paiement, dans le Chapitre 6. Pour gagner en efficacité, nous introduisons le paiement électronique dit "privé" qui caractérise certaines applications spécifiques comme le télépéage, l'abonnement aux transports publics ou encore le rechargement de véhicules électriques. Leur particularité est qu'une même entité joue à la fois le rôle de la banque, qui délivre le moyen de paiement à l'utilisateur, et du marchand auprès duquel il est utilisé. Grâce à notre signature partiellement aveugle, nous proposons alors un nouveau schéma de paiement électronique privé, prouvé sûr. Un même jeton de paiement peut être réutilisé un certain nombre de fois, pour des montants différents, sans risque pour la vie privée des utilisateurs.

Ces résultats ont été publiés à la conférence FC 2016, dans l'article intitulé *"Private eCash in Practice"* [BBD+17].

Enfin, la troisième partie de ce manuscrit est consacrée à une technique d'anonymisation particulière pour des graphes et matrices.

Anonymisation de graphes. Pour valoriser les données qui sont stockées par obligation légale ou encore par nécessité de service, un mécanisme d'anonymisation doit être utilisé. Cependant, obtenir une anonymisation correcte est délicat. Des nombreuses publications de données dites anonymes se sont finalement révélées inadaptées puisque des individus ont pu être identifiés. En effet, même si une base de données peut donner l'impression d'avoir été anonymisée, grâce à des procédés simples comme la suppression des noms ou des numéros identifiants, des informations auxiliaires peuvent suffire à identifier des individus. Il faut donc trouver un bon compromis entre le niveau de protection de la vie privée attendu et la qualité des données obtenues.

Dans le Chapitre 7, nous étudions le cas de la confidentialité différentielle pour les graphes. La confidentialité différentielle garantit que la présence ou l'absence d'un individu ne change pas significativement les données statistiques obtenues, ce qui assure un bon niveau de respect de la vie privée. Pour satisfaire cette propriété, il faut ajouter un bruit calibré selon la sensibilité des données. Nous présentons un mécanisme qui satisfait la confidentialité différentielle tout en optimisant la quantité de bruit ajoutée selon plusieurs sous-ensembles. De cette façon, nous obtenons une meilleure utilité des données finales. En outre, le cas des graphes est particulièrement intéressant. En effet, ils permettent de représenter des données particulières, comme des graphes sociaux ou des données de mobilité par exemple, où un même individu peut agir sur plusieurs valeurs.

Ces résultats ont été publiés dans un brevet puis à la conférence PST 2016, dans l'article *"Edge-Calibrated Noise for Differentially Private Mechanisms on Graphs"* [BCGO16].

Première partie

Préliminaires

Chapitre 1

Préliminaires mathématiques

Dans ce chapitre, nous rappelons les prérequis nécessaires à la compréhension du manuscrit. Tout d'abord, nous présentons quelques structures algébriques utilisées en cryptographie moderne comme les groupes et les corps. Ensuite, nous introduisons la notion de complexité et quelques définitions essentielles. Enfin, nous détaillons la sécurité prouvée, primordiale en cryptographie. Pour cela, nous introduisons les problèmes qualifiés de difficiles ainsi que les différents modèles et types de preuves existants.

Sommaire

1.1 Rappels d'algèbre

Nous rappelons les définitions principales autour des groupes et des corps. Nous introduisons ensuite les courbes elliptiques et applications bilinéaires utilisées en cryptographie.

1.1.1 Groupes et corps

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

Loi associative : pour tout $(g_1, g_2, g_3) \in \mathbb{G}^3$, $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$;

Existence d'un élément neutre : il existe $e \in \mathbb{G}$ tel que, pour tout $g \in \mathbb{G}$, $e \cdot g = g \cdot e = g$;

Existence d'un symétrique : pour tout $g \in \mathbb{G}$, il existe $g' \in \mathbb{G}$ tel que $g \cdot g' = g' \cdot g = e$.

Par la suite, nous utilisons principalement la notation usuelle de la multiplication pour la loi de groupe. Par conséquent, l'élément neutre est noté $1_{\mathbb{G}}$ et le symétrique d'un élément g du

groupe $(\mathbb{G}, \cdot)$ est appelé inverse, noté g^{-1}. Dans le cas d'une notation additive, comme pour les courbes elliptiques ou pour les anneaux, l'élément neutre est noté $O_{\mathbb{G}}$ et le symétrique, appelé opposé, est noté $-g$. En outre, s'il n'y a pas d'ambiguïté, nous désignons simplement le groupe $(\mathbb{G}, \cdot)$ par $\mathbb{G}$ et l'élément neutre par 1.

Définition 2. Soient $\mathbb{G}$ et $\mathbb{H}$ deux groupes.

Homomorphisme. Un *homomorphisme* de groupes $f : \mathbb{G} \to \mathbb{H}$ satisfait, pour tout couple (x, y) dans $\mathbb{G} \times \mathbb{G}$, $f(x \cdot_{\mathbb{G}} y) = f(x) \cdot_{\mathbb{H}} f(y)$.

Isomorphisme. Un *isomorphisme* de groupes est un homomorphisme bijectif.

Définition 3. Soit $\mathbb{G}$ un groupe.

Sous-groupe. Un *sous-groupe* $\mathbb{H}$ de $\mathbb{G}$ est un sous-ensemble $\mathbb{H} \subset \mathbb{G}$ tel que :

Élément neutre : $1_{\mathbb{G}} \in \mathbb{H}$;

Stabilité par produit : pour tout $(h_1, h_2) \in \mathbb{H}^2$, $h_1 \cdot h_2 \in \mathbb{H}$;

Stabilité par inversion : pour tout $h \in \mathbb{H}$, $h^{-1} \in \mathbb{H}$.

Un sous-groupe est donc lui-même un groupe à part entière.

Groupe commutatif. Un groupe $\mathbb{G}$ est dit *commutatif*, ou *abélien*, lorsque sa loi de composition interne est commutative, *i.e.* pour tout $(g_1, g_2) \in \mathbb{G}^2$, $g_1 \cdot g_2 = g_2 \cdot g_1$.

Groupe cyclique. Un groupe $\mathbb{G}$ est dit *cyclique* s'il existe $g \in \mathbb{G}$ tel que, pour tout $h \in \mathbb{G}$, il existe $n \in \mathbb{N}$ tel que $h = g^n$. L'élément g est alors appelé *générateur* du groupe. On notera alors $\langle g \rangle = \mathbb{G}$.

Ordre d'un groupe. Le cardinal de $\mathbb{G}$, noté $|\mathbb{G}|$, est appelé *ordre du groupe*. Un groupe est dit *fini* si son ordre est fini.

Ordre d'un élément. L'*ordre d'un élément* $g \in \mathbb{G}$ est, s'il existe, le plus petit entier positif $n \geq 1$ tel que $g^n = 1_{\mathbb{G}}$. Sinon, g est dit d'ordre infini.

Dans ce mémoire, tous les groupes considérés sont commutatifs et finis.

Théorème 1 (Lagrange)**.** *Pour tout groupe fini $\mathbb{G}$ et tout sous-groupe $\mathbb{H}$ de $\mathbb{G}$, l'ordre de $\mathbb{H}$ divise celui de $\mathbb{G}$.*

Le Théorème 1 implique que l'ordre d'un élément divise l'ordre du groupe. Il permet de montrer que tout groupe d'ordre premier est cyclique et que tout élément différent de l'élément neutre est un générateur. Nous utilisons ce résultat de façon implicite lorsqu'il est nécessaire de choisir un générateur d'un groupe d'ordre premier pour la construction des protocoles.

Définition 4 (Anneau)**.** Un *anneau* $(\mathbb{A}, +, \cdot)$ est composé d'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 :

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

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

Pour un entier $n \in \mathbb{N}$, deux entiers a et b dans $\mathbb{Z}$ sont dits *congrus modulo* n si $b - a$ est un multiple de n. L'ensemble $\{0, \ldots, n-1\}$ des entiers inférieurs à un nombre entier $n \geq 1$, muni de l'addition et de la multiplication modulo n, est un anneau fini qui sera noté $\mathbb{Z}_n$.

Définition 5 (Corps commutatif)**.** Un *corps commutatif* $\mathbb{K}$ est un anneau unitaire commutatif dans lequel tout élément différent de $0_{\mathbb{K}}$ admet un inverse par la loi $\cdot$. Un corps est dit *fini* si son cardinal est fini.

Si p est premier alors $\mathbb{Z}_p$ muni de l'addition et de la multiplication est un corps fini. En outre, pour un corps fini $\mathbb{K}$, l'ensemble $\mathbb{K}^* = \mathbb{K}\backslash\{0_{\mathbb{K}}\}$ muni de la loi $\cdot$ est un groupe cyclique appelé *groupe multiplicatif de* $\mathbb{K}$.

L'*ordre* d'un corps fini est la puissance d'un nombre premier p soit $|\mathbb{K}| = p^n$ où p est appelé *caractéristique* du corps $\mathbb{K}$. Réciproquement, pour tout nombre premier p et tout entier positif n, il existe un unique corps $\mathbb{K}$ – à isomorphisme près – de cardinal p^n, noté $\mathbb{F}_{p^n}$. Ainsi, pour p premier, nous avons $\mathbb{F}_p = \mathbb{Z}_p$. En revanche, si $n > 1$ alors $\mathbb{F}_{p^n} \neq \mathbb{Z}_{p^n}$ car $\mathbb{Z}_{p^n}$ n'est pas un corps.

Application à la cryptographie. La sécurité des protocoles cryptographiques étudiés dans ce mémoire repose sur des problèmes calculatoires difficiles détaillés à la Section 1.3.1. La majorité des problèmes utilisés sont basés sur celui du logarithme discret : calculer l'exponentiation $h = g^x$, pour un générateur g d'un groupe cyclique $\mathbb{G}$, est facile mais l'inverse, retrouver x, est difficile plus l'ordre de g est grand. Nous utilisons cette fonction dite à sens unique (voir Définition 11) dans les groupes d'ordre premier où tous les éléments sont d'ordre maximal – excepté l'élément neutre. L'algorithme de Pohlig-Hellman [PH78] permet de ramener la difficulté du logarithme discret dans un groupe d'ordre n à la difficulté du logarithme discret dans son plus grand sous-groupe d'ordre premier.

En particulier, le groupe multiplicatif $\mathbb{F}_p^*$ est cyclique d'ordre $p-1$. Nous considérons alors un sous-groupe d'ordre q de $\mathbb{F}_p^*$ pour un q premier qui divise $p-1$. Pour résister aux attaques qui utilisent la méthode du calcul d'indice [Adl79], $\mathbb{F}_p^*$ doit être choisi de taille suffisamment grande. De même, la taille du sous-groupe d'ordre q est choisie suffisamment grande pour résister aux algorithmes de résolution du logarithme discret tel que "pas de bébé, pas de géant" [Sha69]. En pratique, il est recommandé de choisir un premier p de longueur au moins 3072 bits et un premier q de taille au moins 256 bits [ANS13]. Aujourd'hui, nous utilisons de plus en plus les groupes de points d'une courbe elliptiques, détaillés ci-après.

1.1.2 Courbes elliptiques et couplages

Courbes elliptiques

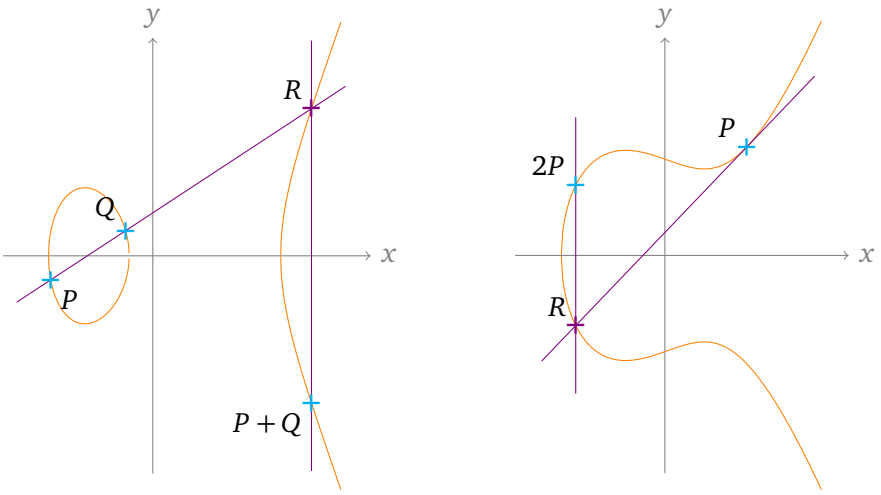
L'utilisation des courbes elliptiques en cryptographie a été introduite indépendamment par MILLER [Mil86] et KOBLITZ [Kob87]. Nous rappelons brièvement la définition et les propriétés principales.

Définition 6 (Courbe Elliptique)**.** Soit p un nombre premier. Une *courbe elliptique* $E(\mathbb{F}_q)$ définie sur $\mathbb{F}_q$, avec $q = p^k$ pour k un entier supérieur à 1, est une courbe non-singulière [1] dont l'ensemble des points $(x, y) \in \mathbb{F}_q^2$ vérifient l'équation de Weierstrass suivante :

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

avec a_1, a_2, a_3, a_4, a_6 dans $\mathbb{F}_q$, complété d'un point dit "à l'infini" $\mathcal{O}$.

1. Le caractère *non-singulier* de la courbe impose que celle-ci ne possède ni point double ni point de rebroussement.



(a) Courbe $y^2 = x^3 - 3x + 1$ (b) Courbe $y^2 = x^3 - x + 2$

FIGURE 1.1 – Exemples d'addition et de doublement de points sur courbes elliptiques

Si la caractéristique p est différente de 2 ou 3 alors il existe une équation courte de la forme

$$y^2 = x^3 + ax + b$$

où $a, b \in \mathbb{F}_q$ tels que $4a^3 + 27b^2 \neq 0$.

Structure de groupe. L'ensemble $E(\mathbb{F}_{q^k})$ peut être muni d'une loi de composition interne notée $+$. Étant donné deux points distincts P et Q de $E(\mathbb{F}_{q^k})$, la droite L passant par ces deux points recoupe la courbe E en un troisième point R de coordonnées (x_R, y_R). La somme $P+Q$ des deux points P et Q est alors définie comme le symétrique de R par rapport à l'axe des abscisses de coordonnées $(x_R, -y_R)$. En outre, si $P = Q$ alors L est définie comme la tangente à la courbe en P. Les Figures 1.1a et 1.1b illustrent respectivement ces deux propriétés.

En considérant le point à l'infini $\mathcal{O}$ comme élément neutre pour cette loi $+$, nous obtenons les règles suivantes :

- si $P = \mathcal{O}$ alors $P + Q = Q$;
- l'opposé du point $P = (x_P, y_P)$ de $E(\mathbb{F}_{q^k})$ est noté $-P$ et a pour coordonnées $(x_P, -y_P)$;
- si $P = -Q$, alors $P + Q = \mathcal{O}$.

Ainsi, muni de la loi de composition interne $+$, l'ensemble $E(\mathbb{F}_{q^k})$ forme un groupe commutatif d'élément neutre $\mathcal{O}$

Pour la cryptographie, l'avantage principal des courbes elliptiques est que, de manière générale, il n'existe pas d'attaques connues autres que les attaques dites génériques. Ces dernières ne reposent pas sur une structure particulière propre au groupe considéré mais fonctionnent pour n'importe quel groupe. Ainsi, pour un même niveau de sécurité, les courbes elliptiques permettent d'utiliser des paramètres plus petits que dans le cas des corps finis. En outre, elles deviennent plus efficaces et donc mieux adaptées aux environnements restreints comme une carte SIM, par exemple. En pratique, il est recommandé de choisir un p de taille au moins 200

bits [ANS13]. Les courbes elliptiques permettent également l'utilisation des groupes bilinéaires et couplages définis ci-après.

Couplages

Un couplage est une application qui permet de transporter un relation entre les points d'une courbe elliptique en une relation équivalente dans le groupe multiplicatif d'un corps fini. En 1991, MENEZES, VANSTONE et OKAMOTO [MVO91] utilisent les couplages pour résoudre le problème du logarithme discret pour certaines familles de courbes. En 2000, JOUX [Jou00] introduit l'utilisation des couplages pour la construction de briques cryptographiques, à travers une variante de l'échange de clés Diffie-Hellman [DH76] entre trois parties. Aujourd'hui, les couplages sont présents dans de nombreux schémas – des signatures ou du chiffrement, par exemple.

Définition 7 (Environnement bilinéaire et couplage)**.** Un *environnement bilinéaire* est l'ensemble des éléments $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ tels que :

- $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ sont trois groupes cycliques d'ordre premier p ;
- g_1 est un générateur de $\mathbb{G}_1$ et g_2 est un générateur de $\mathbb{G}_2$;
- l'application $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, appelée *couplage*, vérifie les propriétés suivantes :

  Bilinéaire : pour tout $(x, y) \in \mathbb{G}_1 \times \mathbb{G}_2$ et tout $(a, b) \in \mathbb{Z}_p^2$, $e(x^a, y^b) = e(x, y)^{ab}$;

  Non-dégénérée : pour tout $(x, y) \in \mathbb{G}_1 \times \mathbb{G}_2$, si $e(x, y) = 1_{\mathbb{G}_T}$ alors $x = 1_{\mathbb{G}_1}$ ou $y = 1_{\mathbb{G}_2}$;

  Efficace : pour tout $(x, y) \in \mathbb{G}_1 \times \mathbb{G}_2$, il existe un algorithme efficace qui calcule $e(x, y)$.

La notion d'algorithme efficace est détaillée à la Section 1.2.1.

À noter que des groupes bilinéaires composites, d'ordre n non premier, existent également. Ils ont été introduits en cryptographie par BONEH, GOH et NISSIM [BGN05] mais impliquent une taille importante et des calculs bien plus complexes que dans le cas des groupes d'ordres premiers.

En pratique, les groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ sont choisis avec précaution. $\mathbb{G}_1$ 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 elliptique implique la sécurité et l'efficacité du couplage. GALBRAITH, PATERSON et SMART [GPS08] ont proposé une classification des couplages selon l'existence, ou non, d'un isomorphisme entre $\mathbb{G}_1$ et $\mathbb{G}_2$ ou inversement :

Type 1 : il existe deux isomorphismes calculables efficacement $\phi_1 : \mathbb{G}_1 \to \mathbb{G}_2$ et $\phi_2 : \mathbb{G}_2 \to \mathbb{G}_1$;

Type 2 : il existe un isomorphisme calculable efficacement $\phi : \mathbb{G}_2 \to \mathbb{G}_1$ mais aucun calculable efficacement n'est connu de $\mathbb{G}_1$ vers $\mathbb{G}_2$;

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

Étant donné que $\mathbb{G}_1$ et $\mathbb{G}_2$ sont cycliques et de même ordre premier, ils sont isomorphes. La sécurité des protocoles cryptographiques ne repose donc pas sur leur existence mais bien sur la difficulté à calculer efficacement ces isomorphismes – dans le cas des types 2 et 3. Les groupes de type 1 sont de moins en moins utilisés en cryptographie car la présence des deux isomorphismes implique plus de possibilité d'attaques pour l'adversaire. Ainsi, certains problèmes jugés difficiles dans les groupes de type 2 et 3 sont rendus faciles dans ceux de type 1.

À noter que le type 1 est parfois présenté avec $\mathbb{G}_1 = \mathbb{G}_2$. Dans tous les cas, les environnements bilinéaires de type 1 sont appelés *symétriques*, par oppostion à ceux de type 2 et 3 qui sont *asymétriques*. Dans le cas symétrique, l'environnement bilinéaire associé est alors noté $(p, \mathbb{G}, \mathbb{G}_T, g, e)$.

1.2 Fonctions de bases

Dans ce mémoire, nous utilisons un ensemble de fonctions classiques en cryptographie dont nous rappelons ici les principales définitions.

1.2.1 Notions de complexité

Tout d'abord, nous donnons quelques bases de la complexité : les notions d'algorithme polynomial, de fonction négligeable et de probabilité négligeable.

Le terme d'algorithme *efficace* correspond à une application qui peut être évaluée en un temps jugé raisonnable en pratique. De façon formelle, nous parlons d'algorithme polynomial défini comme suit.

Définition 8 (Algorithme polynomial)**.** Un algorithme $\mathcal{A}$ est dit *polynomial* ou *s'exécutant en temps polynomial* s'il existe un polynôme p tel que, étant donné une entrée $x \in \{0,1\}^*$, le temps d'exécution est borné par $p(|x|)$ où $|x|$ est la longueur de l'entrée x.

Sauf mention contraire, un *adversaire*, contre un schéma donné, correspond à un algorithme s'exécutant en temps polynomial. En outre, la notion de fonction négligeable est primordiale.

Définition 9 (Fonction négligeable)**.** Une fonction $\nu : \mathbb{N} \to \mathbb{R}^+$ est dite *négligeable* si, pour tout entier positif c, il existe un entier k_c tel que :

$$\nu(k) < \frac{1}{k^c}, \text{ pour tout entier } k \geq k_c.$$

Cette définition nous permet d'introduire celles de probabilité négligeable et de probabilité écrasante.

Définitions 10 (Probabilité négligeable, probabilité écrasante)**.** Soit $\mathbb{P}$ une probabilité dépendant d'un paramètre k, appelé *paramètre de sécurité*.

- $\mathbb{P}$ est dite *négligeable* si $\mathbb{P}$ est une fonction négligeable de k ;
- $\mathbb{P}$ est dite *écrasante* si la probabilité $1 - \mathbb{P}$ est négligeable.

1.2.2 Fonctions particulières

Une fonction à sens unique est facile à calculer dans un sens mais difficile à inverser. Il s'agit d'un outil fondamental en cryptographie.

Définition 11 (Fonction à sens unique)**.** Une fonction $f : \{0,1\}^* \to \{0,1\}^*$ est dite *à sens unique* si elle vérifie les deux propriétés suivantes :

Efficace : il existe un algorithme efficace qui, pour une valeur $x \in \{0,1\}^*$ en entrée, renvoie $f(x)$;

À sens unique : pour tout algorithme efficace et pour tout élément $y = f(x)$, la probabilité de trouver $x' \in \{0,1\}^k$ tel que $y = f(x')$ est négligeable, pour k grand.

Les fonctions de hachage cryptographiques sont un cas particulier des fonctions à sens unique. Entre autres, elles permettent de transformer une chaîne de bits de longueur arbitraire $\{0,1\}^*$ en une chaîne condensée de taille fixe $\{0,1\}^k$ où k est le paramètre de sécurité.

Définition 12 (Fonction de hachage cryptographique)**.** Une *fonction de hachage* $H : \{0,1\}^* \to \{0,1\}^k$, avec k un paramètre de sécurité, est dite *cryptographiquement sûre* si elle vérifie les propriétés suivantes :

Résistance à la pré-image – sens unique : pour tout $y \in \{0,1\}^k$, la probabilité de trouver x tel que $H(x) = y$ est négligeable ;

Résistance à la seconde pré-image : pour $x \in \{0,1\}^*$, la probabilité de trouver $x' \neq x$ tel que $H(x) = H(x')$ est négligeable ;

Résistance aux collisions : la probabilité de trouver un couple $(x, x') \in (\{0,1\}^*)^2$ tel que $H(x) = H(x')$ avec $x \neq x'$ est négligeable.

Des fonctions de hachages cryptographiques sont standardisées comme SHA-256 [SHA02] et SHA-3 [SHA14] qui est recommandée aujourd'hui.

De façon informelle, une fonction est dite pseudo-aléatoire lorsqu'il est difficile de distinguer sa sortie d'une valeur aléatoire.

Définition 13 (Famille de fonctions pseudo-aléatoires)**.** Étant donné λ_1, λ_2 deux fonctions polynomiales et k un paramètre de sécurité, nous notons $\mathcal{R}_k$ l'ensemble des fonctions de $\{0,1\}^{\lambda_1(s)} \to \{0,1\}^{\lambda_2(s)}$. Une famille de fonctions $\mathcal{F}_k = \{F_s\}_{s\in\{0,1\}^k}$ telle que $F_s : \{0,1\}^{\lambda_1(s)} \to \{0,1\}^{\lambda_2(s)}$, est dite *pseudo-aléatoire* si elle vérifie les propriétés suivantes :

Efficace : Étant donné s et x, il existe un algorithme efficace pour évaluer $F_s(x)$.

Pseudo-Aléatoire : pour tout algorithme polynomial $\mathcal{A}$ dont le nombre maximal de requêtes est limité pour $\mathcal{F}_k$ et pour tout k, la différence – appelée *avantage* – entre la probabilité que $\mathcal{A}$ identifie correctement la fonction F_s choisie aléatoirement dans $\mathcal{F}_k$ et la probabilité qu'il identifie correctement R choisie aléatoirement dans $\mathcal{R}_k$ est négligeable,.

Par abus de langage, nous parlons directement de la fonction F_s comme fonction pseudo-aléatoire.

1.3 Sécurité prouvée

Assurer la sécurité des protocoles cryptographiques proposés est aujourd'hui essentielle. Cependant, il n'est pas possible d'obtenir à la fois un schéma efficace et résistant à tout adversaire quelque soit la puissance de ce dernier. Par exemple, le chiffrement dit "parfait" de Vernam, aussi appelé "masque jetable", est inadapté aux cas pratiques puis que la clé utilisée doit être au moins aussi longue que le message à chiffrer et ne peut pas être réutilisée.

Nous nous appuyons donc sur une sécurité dite *calculatoire* qui est basée sur la difficulté de résoudre un problème considéré comme *difficile*. En d'autres termes, si un attaquant est en

mesure de mettre en défaut la sécurité du schéma, c'est qu'il est également à même de résoudre un problème pourtant jugé difficile. Ces problèmes difficiles vont être utilisés pour construire des preuves de la sécurité de certaines propriétés du schéma.

Nous présentons l'ensemble des problèmes difficiles considérés dans ce mémoire. Ensuite, nous détaillons le fonctionnement des différents types et modèles de preuves.

1.3.1 Problèmes difficiles

De façon informelle, un problème est considéré comme difficile lorsque sa résolution est calculatoirement difficile. Autrement dit, pour un problème donné, la probabilité de succès de l'adversaire est négligeable. Chaque problème énoncé ici est associé à une hypothèse qui le suppose difficile. Nous évaluons alors la sécurité des schémas sous une hypothèse particulière. Nous distinguons ici trois catégories de problèmes/hypothèses difficiles.

Problèmes basés sur le logarithme discret

Le problème du logarithme discret est aujourd'hui très largement utilisé et est à l'origine de nombreuses variantes de ce problème. Il repose sur la difficulté de calculer un logarithme discret pour un élément donné, dans un groupe $\mathbb{G}$ contenant un sous-groupe cyclique engendré par g. En effet, la fonction exponentielle qui à un élément $x \in \mathbb{Z}_p$ associe $y = g^x$ est facile à calculer, en utilisant par exemple la méthode dite de *"Square and Multiply"* [Coh93, Section 1.2]. Cependant, dans la majorité des groupes, inverser cette fonction – pour un y donné dans $\mathbb{G}$, trouver le logarithme discret x de y en base g – ne peut pas être réalisé en temps polynomial. C'est, par exemple, le cas pour ceux utilisés en cryptographie comme les groupes multiplicatifs de corps finis ou les courbes elliptiques.

Définition 14 (Problème DL)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p. Étant donné un générateur $g \in \mathbb{G}$ et un élément $h \in \mathbb{G}$, le *problème du logarithme discret* (DL) consiste à calculer $x \in \mathbb{Z}_p$ tel que $h = g^x$. L'entier x est appelé le logarithme discret de h en base g, noté $x = \log_g h$.

En 2003, BELLARE, NAMPREMPRE, POINTCHEVAL et SEMANKO [BNPS03] étendent ce problème au logarithme discret supplémentaire en ajoutant un algorithme efficace, appelé *oracle*, $\mathcal{O}_{\mathsf{DL}}$ auquel l'adversaire a accès. Cet oracle est capable de résoudre le problème du logarithme discret : pour une valeur $h \in \mathbb{G}$ en entrée, il renvoie une valeur $x \in \mathbb{Z}_p$ telle que $h = g^x$.

Définition 15 (Problème OMDL)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p. Étant donné un générateur g et un ensemble de $\ell + 1$ valeurs $(h_1, \ldots, h_\ell, h_{\ell+1}) \in \mathbb{G}^{\ell+1}$, le *problème du logarithme discret supplémentaire* (*one-more discrete logarithm* en anglais, noté OMDL) consiste à renvoyer $\ell + 1$ couples $((x_1, h_1), \ldots, (x_{\ell+1}, h_{\ell+1}))$ tels que $x_i = \log_g h_i$ pour tout $i \in [1, \ell + 1]$ en ayant accès au plus ℓ fois à l'oracle $\mathcal{O}_{\mathsf{DL}}$.

En outre, le problème du logarithme discret possède une large sous-famille basée sur le protocole d'échange de clés de Diffie-Hellman [DH76]. Un problème légèrement plus facile est le problème de Diffie-Hellman calculatoire.

Définition 16 (Problème CDH)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p. Étant donné un générateur $g \in \mathbb{G}$ et deux éléments $(g^a, g^b) \in \mathbb{G}^2$ avec a et b dans $\mathbb{Z}_p$, le *problème Diffie-Hellman calculatoire* (CDH) consiste à calculer g^{ab}.

Une variante décisionnelle de ce problème existe. Dans ce cas, l'adversaire doit décider si les éléments fournis forment un triplet Diffie-Hellman (g^a, g^b, g^{ab}) ou non.

Définition 17 (Problème DDH)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p. Étant donné un générateur $g \in \mathbb{G}$, deux éléments $(A = g^a, B = g^b) \in \mathbb{G}^2$ et un candidat $X \in \mathbb{G}$, le *problème Diffie-Hellman décisionnel* (DDH) consiste à décider si $X = g^{ab}$ est vrai ou non.

Ce problème peut également être présenté sous la forme suivante : étant donné deux générateurs $(g, h) \in \mathbb{G}^2$ et deux éléments $(g^a, h^b) \in \mathbb{G}^2$ avec a et b dans $\mathbb{Z}_p$, décider si $a = b$ ou pas.

En 2001, OKAMOTO et POINTCHEVAL [OP01] introduisent les problèmes dits *gap* où l'accès à un *oracle* pour un problème décisionnel donné est autorisé. Ainsi, le problème *gap* Diffie-Hellman autorise l'accès à l'oracle $\mathcal{O}_{\mathsf{DH}}$ qui renvoie 1 si l'entrée donnée est un triplet DDH et 0 sinon.

Définition 18 (Problème GDH)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p. Étant donné un générateur $g \in \mathbb{G}$ et deux éléments $(A = g^a, B = g^b) \in \mathbb{G}^2$, le *problème gap Diffie-Hellman* (GDH) consiste à calculer l'élément $X = g^{ab} \in \mathbb{G}$ en ayant accès à l'oracle $\mathcal{O}_{\mathsf{DH}}$.

Enfin, la dernière variante considérée est celle du problème d'inversion de DDH présenté ici dans le cas d'un groupe cyclique d'ordre premier [CHL05].

Définition 19 (Problème $q-$DDHI)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p. Étant donné un générateur $g \in \mathbb{G}$, les valeurs $(g, g^x, \ldots, g^{x^q}) \in \mathbb{G}^{q+1}$ pour un entier q donné avec x aléatoire dans $\mathbb{Z}_p$ et un candidat $X \in \mathbb{G}$, le *problème d'inversion q—Diffie-Hellman décisionnel* ($q-$DDHI) consiste à décider si $X = g^{1/x}$ est vrai ou non.

Problèmes bilinéaires

Dans le cas des schémas basés sur des applications bilinéaires, des problèmes difficiles spécifiques existent. Il s'agit à nouveau de variantes des problèmes classiques mais adaptées à l'environnement considéré.

Il est à noter que le problème DDH est plus facile que le problème CDH. Ainsi, un algorithme capable de résoudre CDH sera en mesure de résoudre DDH. Par exemple, dans le cas d'un environnement bilinéaire $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ de type 1, le problème DDH est facile tandis que CDH reste difficile. En effet, pour tester si $c = ab$, il suffit de vérifier si $e(g^a, g^b) = e(g^c, g)$. Pour d'autres types de couplages, le problème DDH reste difficile étant donné leur asymétrie. En 2004, BONEH, BOYEN et SACHAM [BBS04] formalisent l'hypothèse dite de Diffie-Hellman externe (XDH) qui repose sur l'hypothèse DDH, dans le cas particulier de deux groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ distincts où le problème DDH est supposé difficile dans $\mathbb{G}_1$.

Définition 20 (Hypothèse XDH)**.** Soit $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ un environnement bilinéaire avec p premier, sans isomorphisme calculable efficacement de $\mathbb{G}_1$ vers $\mathbb{G}_2$. L'*hypothèse Diffie-Hellman eXterne* (XDH) suppose que le problème DDH est difficile dans le groupe $\mathbb{G}_1$.

En 2004, BONEH et BOYEN [BB04] introduisent le problème $q-$SDH pour prouver la sécurité de leur schéma de signature – détaillé à la Section 2.2.1.

Définition 21 (Problème $q-\mathsf{SDH}$)**.** Soit $(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,g_1,g_2)$ un environnement bilinéaire avec p premier. Étant donné $(g_1,g_1^x,g_1^{x^2}\ldots,g_1^{x^q},g_2,g_2^x)\in\mathbb{G}_1^{q+1}\times\mathbb{G}_2^2$ avec x dans $\mathbb{Z}_p$, le *problème q—Diffie-Hellman flexible* ($q-\mathsf{SDH}$) consiste à trouver un couple $(c,g_1^{\frac{1}{x+c}})\in\mathbb{Z}_p\times\mathbb{G}_1$.

Le problème $q-\mathsf{SDH}$ reste difficile y compris dans les groupes où il existe un algorithme efficace $\mathcal{O}_{\mathsf{DH}}$ qui, pour une entrée (g,h,g^a,h^b), renvoie 1 si $a=b \mod p$ et 0 sinon. Plus formellement, ces groupes sont qualifiés de groupes gap-DDH et $\mathcal{O}_{\mathsf{DH}}$ est un oracle. Par définition, un groupe cyclique $\mathbb{G}$ muni d'une application bilinéaire e de type 1 est un groupe gap-*DDH* puisqu'il est facile de vérifier si (g,h,g^a,h^b) est un quadruplet Diffie-Hellman en vérifiant si $e(g^a,h)=e(g,h^b)$.

En 2009, FUCHSBAUER, POINTCHEVAL et VERGNAUD [FPV09] prouvent que l'hypothèse $q-$SDH dans un groupe gap-DDH implique l'hypothèse gap $q-\mathsf{SDH}-\mathsf{III}$ définie ci-après.

Définition 22 (Problème *gap* $q-\mathsf{SDH}-\mathsf{III}$)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p. Étant donné $(g,h,g^y)\in\mathbb{G}^3$ et q triplets distincts $(x_i,y_i,A_i=(g^{y_i}h)^{\frac{1}{\gamma+x_i}})\in\mathbb{Z}_p^2\times\mathbb{G}$ avec x_i et z_i aléatoires dans $\mathbb{Z}_p$ pour $i\in[1,q]$, le *problème (III) gap q-Diffie-Hellman flexible* ($q-\mathsf{SDH}-\mathsf{III}$) consiste à trouver un nouveau triplet $(x,y,A=(g^yh)^{\frac{1}{\gamma+x}})$ pour x et z dans $\mathbb{Z}_p$, en ayant accès à l'oracle $\mathcal{O}_{\mathsf{DH}}$.

Autres problèmes

En 1999, LYSYANSKAYA, RIVEST, SAHAI et WOLF [LRSW99] introduisent un problème, noté LRSW selon leurs initiales, pour prouver la sécurité de leur schéma.

Définition 23 (Problème LRSW)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p, généré par g. Pour $(X=g^x,Y=g^y)\in\mathbb{G}^2$ avec x et y aléatoires dans $\mathbb{Z}_p$, l'oracle $\mathcal{O}_{\mathsf{LRSW}}^{(X,Y)}$ prend en entrée $m\in\mathbb{Z}_p$ et renvoie le triplet $A=(a,a^y,a^{x+mxy})$ avec a tiré aléatoirement dans $\mathbb{G}$. Étant donné le couple (X,Y) et un accès sans restrictions à $\mathcal{O}_{\mathsf{LRSW}}^{(X,Y)}$, le *problème* LRSW consiste à trouver un triplet $(b,b^y,b^{x+m'xy})$ pour un message m' qui n'a jamais été soumis à l'oracle.

En 2015, POINTCHEVAL et SANDERS [PS16] introduisent une variante de l'hypothèse LRSW qui repose sur le problème précédent, dans le cas particulier des groupes bilinéaires de type 3. Cette hypothèse a été prouvée sous le modèle dit du groupe générique, détaillé à la Section 1.3.2.

Définition 24 (Hypothèse Assumption1)**.** Soit $(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,g_1,g_2)$ un environnement bilinéaire de type 3. Pour $(X_1=g_1^x,Y_1=g_1^y)$ et $(X_2=g_2^x,Y_2=g_2^y)$ avec x et y tirés aléatoirement dans $\mathbb{Z}_p$, l'oracle $\mathcal{O}_{\mathsf{Assumption1}}^{(X_1,Y_1,X_2,Y_2)}$ prend en entrée $m\in\mathbb{Z}_p$ et renvoie le couple $P=(h,h^{x+my})$ avec h aléatoire dans $\mathbb{G}_1$. Étant donné (g_1,Y_1,g_2,X_2,Y_2) et un accès sans restrictions à $\mathcal{O}_{\mathsf{Assumption1}}^{(X_1,Y_1,X_2,Y_2)}$, l'*hypothèse* appelée Assumption1 suppose qu'il est difficile de générer un nouveau couple valide pour un message m' qui n'a jamais été soumis à l'oracle, avec $h\neq 1_{\mathbb{G}_1}$.

Pour finir, la dernière définition qui nous sera utile est celle de l'hypothèse décisionnelle du résidu composite. Introduite en 1999 par PAILLIER [Pai99], elle permet de prouver la sécurité de son schéma de chiffrement – voir Section 2.1.2. Elle repose sur la difficulté de déterminer si un élément de $\mathbb{Z}_{n^2}^*$ est une puissance n-ième modulo n^2 ou non.

Définition 25 (Hypothèse DCR)**.** Soit un entier $n \in \mathbb{Z}$. Étant donné $z \in \mathbb{Z}_{n^2}$, l'*hypothèse décisionnelle du résidu composite* suppose qu'il est difficile de distinguer $\mathbb{Z}_{n^2}^n$ de $\mathbb{Z}_{n^2}^*$ où $\mathbb{Z}_{n^2}^n = \{z \in \mathbb{Z}_{n^2}^* \mid \exists y \in \mathbb{Z}_{n^2}^* : z = y^n \bmod n^2\}$, autrement dit, de déterminer s'il existe ou non un $y \in \mathbb{Z}_{n^2}^*$ tel que $z \equiv y^n \bmod n^2$.

1.3.2 Types de preuves et modèles de sécurité

Pour attester de la robustesse d'un schéma, sa sécurité doit donc être prouvée. Pour cela, il est nécessaire de montrer qu'il est difficile, au sens calculatoire, de mettre en défaut sa sécurité. Nous utilisons alors des preuves de sécurité réalisées dans un modèle particulier.

Types de preuves - sécurité par réduction

Nous présentons ici trois types de preuves différentes. Derrière chacune de ces méthodes, nous retrouvons l'idée de sécurité dite *réductionniste*.

Preuves standard, par réduction. Dès 1984, GOLDWASSER et MICALI [GM84] introduisent les bases de la sécurité prouvée réductionniste. Ces preuves par réduction reposent sur l'idée que si un adversaire parvient à *casser* la sécurité du schéma alors il est parvenu à résoudre un problème considéré comme difficile – ou à casser la propriété d'un autre schéma lui-même prouvé sûr. Pour formaliser cela, nous construisons un adversaire $\mathcal{A}$ contre une expérience $\mathrm{Exp}_{\mathcal{A}}^{p}$ propre à une propriété p. De plus, un challenger $\mathcal{C}$ est, quant à lui, construit contre le problème difficile. Pour modéliser les capacités de l'adversaire, nous lui donnons éventuellement accès à un certains nombres d'oracles. Indirectement, l'adversaire est utilisé pour casser le problème difficile – en ajoutant une interface entre $\mathcal{A}$ et $\mathcal{C}$. Ses réponses contre l'expérience $\mathrm{Exp}_{\mathcal{A}}^{p}$ sont utilisés par le challenger $\mathcal{C}$ pour casser le problème difficile. Bien sûr, casser le problème difficile n'est possible qu'avec une probabilité négligeable. Cela nous permet d'affirmer que la probabilité de succès de l'adversaire est négligeable également et donc que le schéma est sûr. Nous parlerons également d'*avantage* négligeable de l'adversaire. Cela correspond la probabilité de succès relative à l'expérience $\mathrm{Exp}_{\mathcal{A}}^{p}$ considérée.

Preuves par jeux. Une autre méthode pour réaliser des preuves par réduction consiste à utiliser une suite d'étapes appelée séquence de *jeux* [BG90, KR96, BR04]. Formalisée en 2004 par SHOUP [Sho04], cette technique consiste tout d'abord à décrire formellement l'attaque réalisée par l'adversaire, sous la forme d'une expérience. Cette description est alors appelée *jeu 0* et est associée à l'évènement "S_0 : l'adversaire gagne". Ensuite, une séquence de jeux est définie où chaque jeu i est associé à un évènement "S_i : l'adversaire gagne". La preuve de sécurité consiste alors à montrer que la probabilité de succès $\mathbb{P}(S_i)$ au jeu i est très proche de la probabilité de succès $\mathbb{P}(S_{i+1})$ à l'étape $i+1$. Le but est de créer autant d'instance que nécessaire pour se ramener à un jeu n où la probabilité est facile à calculer. Au final, la probabilité de succès de l'adversaire $\mathbb{P}(S_0)$ à l'étape initiale sera alors très proche de la probabilité de succès $\mathbb{P}(S_n)$ au jeu n.

Preuves par simulation. Les preuves par simulation sont particulièrement adaptées aux schémas multipartites et reposent sur le paradigme "monde idéal/monde réel". Le but de

la preuve de sécurité est alors de montrer que, du point de vue d'un adversaire, ces deux mondes sont indistinguables. Le monde dit réel correspond simplement au schéma étudié. En revanche, dans le monde idéal, il est considéré qu'aucune attaque n'est possible. Une fonctionnalité idéale $\mathcal{F}$ est définie : elle est exécutée par un tiers de confiance dont le rôle est central puisqu'il permet aux participants d'interagir entre eux. Le protocole sera alors dit sûr si la vue d'un adversaire pendant l'exécution peut être simulée par un algorithme, appelé *simulateur*, qui interagit exclusivement avec le tiers de confiance du monde idéal. Cette méthode de preuves suit le principe de composition séquentielle c'est-à-dire qu'il est possible d'utiliser un ensemble de protocoles, prouvés sûrs indépendamment selon ce paradigme, pour construire un système plus complexe où la sécurité sera assurée par "composition". Pour plus de détails, nous renvoyons le lecteur au tutoriel proposé par LINDELL [Lin16].

Modèle de sécurité

Un *modèle de sécurité* définit l'environnement dans lequel l'adversaire réalise son attaque. Nous distinguons ici trois modèles majeurs.

Modèle de l'oracle aléatoire. En 1986, FIAT et SHAMIR [FS87] introduisent les oracles aléatoires dont le modèle est formalisé en 1993 par BELLARE et ROGAWAY [BR93]. Le *modèle de l'oracle aléatoire* consiste à considérer les fonctions de hachage comme des fonctions parfaitement aléatoires. En pratique, la fonction de hachage prend alors la forme d'un oracle. Chaque fois qu'un adversaire a besoin du résultat de la fonction de hachage sur une entrée de son choix, il appelle l'oracle aléatoire qui va lui retourner une valeur indistinguable de celle qu'aurait produite une fonction parfaitement aléatoire et déterministe sur la même entrée.

Ce modèle est parfois critiqué car il est jugé trop idéalisé. En particulier, des schémas [CGH98, CGH04, BB04] mettent en avant la fragilité de ce modèle. Néanmoins, cela concerne des constructions très singulières. Le modèle de l'oracle aléatoire reste donc très utilisé en pratique car il permet, grâce à cette idéalisation de la fonction de hachage, la construction de schémas efficaces.

Modèle du groupe générique. En 1994, NECHAEV [Nec94] introduit le *modèle du groupe générique* [Sho97, Mau05, JS08]. Il s'agit d'un autre modèle "idéalisé" qui suppose que les opérations arithmétiques n'ont pas de propriétés particulières qui pourraient être utilisées par un attaquant pour obtenir de l'information. Autrement dit, l'adversaire ne peut pas exploiter les propriétés de structure de groupe pour réaliser ses attaques. Pour chaque opération de groupe qu'il souhaite effectuer – une addition dans le contexte des courbes elliptiques ou une multiplication par exemple, il doit faire appel à un oracle.

Modèle standard. Contrairement aux cas précédents, le *modèle standard* n'a recours à aucune idéalisation. Seules les propriétés inhérentes aux constructions utilisées sont considérées. Avec ce modèle, la sécurité des constructions ne repose que sur la difficulté calculatoire de certains problèmes mathématiques tels que le problème du logarithme discret énoncé ci-dessus. Cependant, les schémas prouvés sûrs avec ce modèle sont souvent moins efficaces et les preuves sont plus complexes.

De manière générale, pour prouver la sécurité de nos schémas nous suivrons les étapes suivantes :

1. définitions des propriétés de sécurité attendues du schéma ;
2. description du schéma et des hypothèses calculatoires considérées ;
3. constructions des preuves de sécurité selon le modèle et le type de preuves choisis.

Conclusion

Dans ce premier chapitre, nous avons étudié les outils mathématiques de base et le fonctionnement de la sécurité prouvée. Ces notions préliminaires sont indispensables pour la compréhension du mémoire.

Nous allons maintenant nous intéresser plus particulièrement aux différentes primitives cryptographiques qui répondent, entre autres, aux besoins de confidentialité, d'authentification et d'intégrité des données ainsi qu'aux propriétés de respect de la vie privée.

Chapitre 2

Outils cryptographiques

Dans ce chapitre, nous présentons les outils cryptographiques nécessaires à la compréhension de la suite du manuscrit. Tout d'abord, nous rappelons le fonctionnement de la cryptographie dite "à clé publique" dans le contexte de la confidentialité. Ensuite, nous définissons les techniques permettant d'assurer l'authentification et l'intégrité des données, à savoir les signatures numériques et les codes d'authentification de messages. Enfin, nous présentons quelques primitives additionnelles utilisées couramment en cryptographie.

Dans chaque section, nous précisons les définitions générales des méthodes citées, leur sécurité et les schémas particuliers utilisés pour la construction de nos systèmes présentés dans les chapitres suivants.

Sommaire

2.1 Confidentialité

Le but premier de la cryptographie est d'assurer la confidentialité des données. Plus précisément, l'objectif est qu'un individu puisse envoyer un message à une ou plusieurs personnes tout en étant certain que seuls les destinataires légitimes pourront accéder à son contenu. Pour cela, les schémas de chiffrement rendent le message inintelligible pour toute personne extérieure, sous certaines hypothèses.

Historiquement, les premiers schémas introduits sont ceux dit "symétriques" ou "à clé secrète" : l'émetteur et le destinataire des messages se mettent d'accord au préalable sur un secret commun, la clé, qui leur permet de chiffrer et déchiffrer leurs échanges. Ces schémas

restent aujourd'hui largement utilisés en pratique car ils sont très efficaces même sur de gros volumes de données. Néanmoins, pour leur utilisation, il faut que les individus soient capables de s'échanger la clé secrète en toute sécurité.

2.1.1 Chiffrement à clé publique

En réponse au problème de l'échange de clé des schémas symétriques, DIFFIE et HELLMAN [DH76] ont révolutionné la cryptographie avec le concept de cryptographie à "clé publique" ou "asymétrique". Ici, un même utilisateur détient une paire de clés : une clé publique, distribuée à tous, qui permet de chiffrer un message lui étant destiné et une clé privée, connue uniquement par l'utilisateur, qui est utilisée pour déchiffrer les messages reçus de la part de différents expéditeurs.

Dans le cadre de ce mémoire, nous introduisons uniquement les schémas de chiffrement à clé publique. Cependant, il est important de préciser que les deux concepts, symétriques et asymétriques, sont complémentaires. En effet, bien qu'étant plus coûteux, l'algorithme de chiffrement à clé publique peut être utilisé pour échanger la clé secrète qui sera utilisée par un algorithme de chiffrement symétrique pour transmettre efficacement des messages plus longs.

Assurer la confidentialité des données est le rôle principal du schéma de chiffrement. Les deux propriétés attendues d'un tel schéma sont les suivantes :

- un message correctement chiffré avec une clé publique doit pouvoir être correctement déchiffré avec la clé privée associée ;
- à partir du message chiffré, aucune information ne peut être déduite sur le message initial sans la connaissance de la clé privée.

Par abus de langage, nous utiliserons le terme "chiffré" pour désigné un message chiffré.

Plus formellement, la définition du chiffrement à clé publique est la suivante.

Définition 26 (Schéma de chiffrement à clé publique.)**.** Un *schéma de chiffrement à clé publique* est constitué des quatre étapes associées aux algorithmes suivants :

Initialisation. L'initialisation du système est réalisée par l'algorithme `Setup`. Il prend en entrée 1^λ où λ est un paramètre de sécurité et retourne les paramètres publics du système, notés pp.

$$pp \leftarrow \texttt{Setup}(1^\lambda)$$

Génération de clés. L'algorithme de génération des clés est probabiliste et noté `KeyGen`. Il prend en entrée les paramètres pp et génère une paire (pk, sk) contenant la clé publique et la clé privée.

$$(pk, sk) \leftarrow \texttt{KeyGen}(pp)$$

Chiffrement. L'algorithme de chiffrement, déterministe ou probabiliste, est noté `Encrypt`. Il prend en entrée les paramètres pp, un message m à chiffrer et la clé publique pk. Il retourne un chiffré c du message m.

$$c \leftarrow \texttt{Encrypt}(pp, m, pk)$$

Déchiffrement. L'algorithme de déchiffrement est déterministe et noté `Decrypt`. Il prend en entrée les paramètres pp, un chiffré c et la clé privée sk. Il retourne le message clair

m correspondant au chiffré c.

$$m \leftarrow \texttt{Decrypt}(pp, c, sk)$$

Pour être considéré comme sûr, un schéma de chiffrement doit résister à certaines attaques. Celles-ci sont caractérisées par les objectifs de l'adversaire et les moyens dont il dispose. L'**objectif** principal d'un adversaire est de retrouver le message clair. Dans ce cas, les différents buts visés sont les suivants :

Cassage total (UBK) **:** à partir de la clé publique, l'adversaire déduit la clé privée de l'utilisateur et peut alors déchiffrer tous les messages ; la propriété associée est notée UBK pour *Unbreakability* ;

Cassage de la fonction de déchiffrement (OW) **:** sans connaître la clé de l'utilisateur, l'adversaire retrouve le message clair à partir du chiffré – il attaque ainsi la propriété de fonction à sens-unique du schéma de chiffrement ; la propriété associée est alors notée OW pour *One-Wayness*.

À défaut d'obtenir le message clair, l'adversaire peut avoir des objectifs plus modestes. Ainsi, il peut tout simplement essayer d'obtenir de l'information sur le message en cassant les propriétés suivantes :

Non-malléabilité (NM) **:** à partir d'un chiffré c sur un message m qu'il ne connaît pas, l'adversaire génère un nouveau chiffré c' sur un message m' inconnu également mais de telle sorte que m et m' restent liés – par exemple, $m' = m + 1$;

Indistinguabilité (IND) **:** étant donné deux messages, l'adversaire trouve auquel de ces deux messages correspond un chiffré donné, avec une probabilité supérieure à $1/2$ – la propriété d'indistinguabilité est équivalente à celle de la *sécurité sémantique* où un adversaire détermine des informations sur le clair à partir du chiffré [GM84].

Pour atteindre ces objectifs, l'adversaire peut avoir accès à différents types de ressources. Les **moyens** dont il dispose caractérisent sa puissance. Les hypothèses les plus courantes pour un adversaire sont les suivantes :

Attaque à clairs choisis (*Chosen Plaintext Attack* (CPA)) **:** l'adversaire peut obtenir les chiffrés correspondant aux messages clairs de son choix ; en cryptographie asymétrique, l'adversaire peut toujours réaliser cette attaque à partir de la clé publique ;

Attaque à chiffrés choisis (*Chosen Ciphertext Attack* (CCA1)) **:** l'adversaire a accès à un oracle de déchiffrement pendant une période donnée, cela lui permet d'obtenir les messages clairs correspondants aux chiffrés de son choix ;

Attaque à chiffrés choisis adaptative (*Adaptative Chosen Ciphertext Attack* (CCA2)) **:** l'adversaire a accès à un oracle de déchiffrement tout au long de son attaque, cela lui permet d'obtenir les messages clairs correspondants aux chiffrés de son choix et d'adapter ses requêtes à tout moment.

Pour définir la sécurité d'un schéma de chiffrement, il faut prendre en compte ces deux composantes qui caractérisent l'adversaire. Chaque schéma est présenté comme résistant à une attaque définie selon le couple (objectif, moyen) de l'adversaire. En utilisant nos définitions précédentes, l'objectif appartient à l'ensemble {UBK, OW, NM, IND} et le moyen utilisé est compris dans {CPA, CCA1, CCA2}. Il est à noter que des relations entre ses différentes propriétés existent et que des transformations sont parfois possibles [BDPR98, BS99].

2.1.2 Exemples de schémas de chiffrement à clé publique

Pour la construction de nos systèmes, nous utiliserons principalement les schémas de chiffrement à clé publique suivants : le cryptosystème proposé par ELGAMAL et une de ses variantes ainsi que le cryptosystème introduit par PAILLIER.

Cryptosystème ElGamal

Introduit en 1984, le cryptosystème ElGamal [ElG84] est le premier schéma fondé sur le problème du logarithme discret. Plus précisément, ce schéma est sémantiquement sûr (IND-CPA) sous l'hypothèse de Diffie-Hellman décisionnelle (DDH), dans le modèle standard. Nous décrivons ici son fonctionnement.

- $\texttt{Setup}(1^\lambda)$ génère les paramètres publics $pp = (p, \mathbb{G}, g)$ où $\mathbb{G} = \langle g \rangle$ est un groupe cyclique d'ordre premier p où DDH est supposé difficile ;
- $\texttt{KeyGen}(pp)$ choisit aléatoirement une clé privée x dans $\mathbb{Z}_p^*$ et calcule la clé publique associée $h = g^x$;
- $\texttt{Encrypt}(pp, m, h)$ choisit aléatoirement r dans $\mathbb{Z}_p^*$ et calcule :

$$c_1 = g^r \quad \text{et} \quad c_2 = m \cdot h^r.$$

  Le chiffré de m est alors $c = (c_1, c_2)$;
- $\texttt{Decrypt}(pp, c, x)$ retourne le message clair en calculant :

$$m = c_2 \cdot c_1^{-x}.$$

De plus, le chiffrement ElGamal satisfait la propriété dite d'*homomorphisme multiplicatif*. Ainsi, multiplier deux chiffrés permet d'obtenir un nouveau chiffré où le message clair est le produit des deux messages clairs initiaux. En effet, étant donné deux chiffrés $C = (c_1, c_2)$ et $C' = (c_1', c_2')$ pour deux messages m et m' respectivement, la multiplication nous donne un nouveau chiffré $C'' = (c_1'', c_2'')$ où $c_1'' = c_1 c_1' = g^{r_1} g^{r_1'} = g^{r_1 + r_1'}$ et $c_2'' = c_2 c_2' = m h^r \cdot m' h^{r'} = mm' \cdot h^{r+r'}$ pour le message mm'.

Plusieurs variantes du chiffrement ElGamal existent. Dans la suite du manuscrit, nous en utiliserons une en particulier décrite ci-après.

Version modifiée de ElGamal. En 2005, JUELS, CATALANO et JAKOBSSON [JCJ10] proposent une variante du cryptosystème ElGamal, notée M-ElGamal et basée sur les travaux de [JL00, CS98]. Elle reste sémantiquement sûre sous l'hypothèse DDH mais est cette fois-ci résistante aux attaques adaptatives.

- $\texttt{Setup}(1^\lambda)$ génère les paramètres publics $pp = (p, \mathbb{G})$ où $\mathbb{G}$ est un groupe cycle d'ordre premier p.
- $\texttt{KeyGen}(pp)$ choisit aléatoirement la clé privée $sk = (x_1, x_2)$ dans $(\mathbb{Z}_p^*)^2$ et calcule la clé publique associée $pk = (g_1, g_2, h)$ où g_1, g_2 sont des générateurs de $\mathbb{G}$ et $h = g_1^{x_1} g_2^{x_2} \in \mathbb{G}$.
- $\texttt{Encrypt}(pp, m, pk)$ choisit aléatoirement r dans $\mathbb{Z}_p^*$ et calcule :

$$c_1 = g_1^r, \quad c_2 = g_2^r \quad \text{et} \quad c_3 = m \cdot h^r.$$

Le chiffré de m est alors (c_1, c_2, c_3).

– `Decrypt`$(pp, (c_1, c_2, c_3), x)$ retourne le message clair en calculant :

$$m = \frac{c_3}{c_1^{x_1} c_2^{x_2}}.$$

Cryptosystème de Paillier

Proposé en 1999, le cryptosystème de Paillier [Pai99] repose sur le problème des résidus quadratiques (DCR). Il est sémantiquement sûr sous cette hypothèse et est défini ci-après.

– `KeyGen`() choisit aléatoirement deux grand entiers premiers p et q tels que [1] $\text{pgcd}(pq, (p-1)(q-1)) = 1$; il calcule ensuite $n = pq$ et $K = \text{ppcm}(p-1, q-1)$; la clé privée sk est égale à K et la clé publique associée pk est (n, g), avec g dans $\mathbb{Z}_{n^2}^*$;

– `Encrypt`(m, pk) choisit aléatoirement r dans $\mathbb{Z}_n^*$ et calcule le chiffré

$$c = g^m r^n \bmod n^2;$$

– `Decrypt`(c, sk) calcule le message

$$m = \frac{L(c^{sk} \bmod n^2)}{L(g^{sk} \bmod n^2)} \bmod n$$

où L est une fonction définie sur $S_n = \{u \mid u < n^2, u = 1 \bmod n\}$ telle que $L(u) = \frac{u-1}{n}$.

Ce schéma vérifie la propriété dite d'*homomorphisme additif*. Ainsi, multiplier deux chiffrés permet d'obtenir un nouveau chiffré où le message clair est l'addition des clairs initiaux. En effet, étant donné deux chiffrés $c_1 = g^{m_1} r_1^n \bmod n^2$ et $c_2 = g^{m_2} r_2^n \bmod n^2$, la multiplication donne un nouveau chiffré $c = c_1 \cdot c_2 = g^{m_1+m_2}(r_1 r_2)^n \bmod n^2$ pour le message $(m_1 + m_2)$. D'autres égalités intéressantes découlent de cette propriété :

– il est possible d'obtenir un chiffré de $(m_1 + m_2)$ en multipliant uniquement par $g^{m_2} \bmod n^2$ au lieu du chiffré entier de m_2, nous obtenons alors $c = c_1 \cdot g^{m_2} = g^{m_1+m_2} r_1^n \bmod n^2$;

– un chiffré $c = g^m r^n \bmod n^2$ élevé à la puissance donne le chiffré multiplié par une constante c'est-à-dire $c^k = g^{km} r^n \bmod n^2$.

Chiffrement à seuil

Des chiffrements à clé publique, tels que ElGamal et Paillier, peuvent s'adapter à la cryptographie dite à seuil [Des88]. Dans ce cas, la clé privée n'est plus la responsabilité d'une personne unique mais elle est répartie parmi un nombre prédéfini d'individus. En tant qu'utilisateur d'un système, cela évite par exemple de devoir faire confiance à une seule entité et de se prémunir d'attaques ciblées sur le détenteur de la clé. En effet, dans le cas des chiffrements à seuil, les clés sont générées de manière collaborative par n participants. La clé privée est alors partagée entre chaque participant. Pour déchiffrer les messages reçus, il est nécessaire qu'au moins t participants parmi les n coopèrent. La valeur t est alors appelée seuil du cryptosystème, elle détermine le nombre requis minimum de parties de la clé (et donc de participants) pour

1. En choisissant p plus petit que q tels que $\text{pgcd}(pq, (p-1)(q-1)) = 1$, nous nous assurons que p ne divise pas $q-1$. En pratique, c'est toujours vrai étant donné que p et q sont de même longueur.

pouvoir déchiffrer. Pour plus de détails, nous renvoyons le lecteur aux références suivantes [DF90, FS01].

2.2 Authentification et intégrité

Comme nous l'avons vu, le but premier de la cryptographie est la confidentialité des messages. À l'ère de la cryptographie moderne, cela n'est plus suffisant. Il est nécessaire de construire des mécanismes pour assurer à la fois l'authentification et l'intégrité des messages. L'authentification garantit l'identité de l'expéditeur du message. L'intégrité, quant à elle, permet au destinataire d'être certain que le message reçu n'a pas été altéré pendant le transport.

Plusieurs mécanismes répondent à ces problématiques aussi bien dans les constructions symétriques qu'asymétriques. Nous détaillons principalement le cas asymétrique à travers les signatures numériques et certaines de leurs variantes. Ensuite, nous présentons rapidement le cas symétrique avec l'utilisation de codes d'authentification de messages.

2.2.1 Signatures numériques

Tout comme pour les documents papiers, la signature numérique a pour rôle d'authentifier l'auteur d'un message. En outre, elle garantit aussi que ce message n'a pas été modifié depuis la signature. Elle repose sur la cryptographie à clé publique. Chaque utilisateur détient donc une paire de clés : une clé privée, connue uniquement par le signataire, et une autre publique, connue de tous. La clé privée permet de générer la signature qui sera vérifiable par tous, grâce à la clé publique. La signature numérique assure à la fois la vérification publique de l'intégrité du message et celle de l'identité du signataire.

Pour un tel mécanisme, plusieurs propriétés sont attendues :
- l'identité du signataire doit pouvoir être retrouvée avec certitude ;
- un attaquant ne peut pas se faire passer pour le signataire en signant à sa place ;
- la signature ne doit pas être réutilisée pour un autre contenu ;
- une fois signé, il doit être impossible de modifier le document ;
- enfin, le signataire ne doit pas pouvoir répudier une de ces signataire.

La définition plus formelle de la signature numérique est la suivante.

Définition 27 (Schéma de signature numérique.)**.** Un *schéma de signature numérique* est constitué de quatre étapes associées aux algorithmes suivants :

Initialisation. L'initialisation du système est réalisée par l'algorithme `Setup`. Il prend en entrée 1^λ où λ est un paramètre de sécurité et retourne les paramètres publics du système, notés pp.

$$pp \leftarrow \texttt{Setup}(1^\lambda)$$

Génération des clés. L'algorithme de génération des clés est probabiliste et noté `KeyGen`. Il prend en entrée les paramètres pp et génère une paire (pk, sk) contenant la clé publique et la clé privée du signataire.

$$(pk, sk) \leftarrow \texttt{KeyGen}(pp)$$

Signature. L'algorithme de signature est généralement probabiliste et noté `Sign`. Il prend en entrée les paramètres publics pp, un message m à signer et la clé privée sk du signataire. Il retourne une signature σ du message m.

$$\sigma \leftarrow \texttt{Sign}(pp, m, sk)$$

Vérification. L'algorithme de vérification de la signature est déterministe et noté `Verif`. Il prend en entrée les paramètres pp, une signature σ d'un message m donné et la clé publique pk du signataire. Il retourne 1 si la signature est valide et 0 sinon.

$$\{0,1\} \leftarrow \texttt{Verif}(pp, m, \sigma, pk)$$

Comme pour les schémas de chiffrement, un schéma de signature est considéré comme sûr s'il résiste à certaines attaques qui sont caractérisées par les objectifs et les moyens dont dispose l'adversaire. Ici, son objectif principal est de signer des messages à la place du signataire. Les principaux buts sont alors les suivants :

Cassage total (UBK) **:** l'adversaire déduit la clé privée du signataire et peut signer en son nom par la suite – la propriété associée est notée UBK pour *Unbreakability* ;

Falsification universelle (UF) **:** l'adversaire connaît un algorithme équivalent à la signature, il peut alors signer n'importe quel message sans connaître la clé privée – la propriété associée est notée UF pour *Universal Unforgeability* ;

Falsification sélective (SUF) **:** l'adversaire réussit à générer une signature pour un message donné au préalable – la propriété associée est notée SUF pour *Selective Unforgeability* ;

Falsification existentielle (EUF) **:** l'adversaire réussit à générer une signature valide pour au moins un message – qui peut cependant être dénué de sens ; la propriété associée est notée EUF pour *Existential Unforgeability*.

Pour atteindre ces objectifs, l'adversaire peut avoir accès à différentes ressources. Selon sa puissance, les quatre principaux types d'attaques sont :

Attaque sans message (*No Message Attack* (NM)) **:** l'adversaire n'a accès qu'aux paramètres publics et à la clé publique du signataire attaqué ;

Attaque à messages connus (*Known Message Attack* (KMA)) **:** en plus des données publiques, l'adversaire connaît une liste de couples (message, signature) associés à la clé publique du signataire ;

Attaque sélective à message choisis (*Selective Chosen Message Attack* (SCMA)) **:** tant qu'il ne connaît pas la clé publique, l'adversaire a accès à un oracle de signature pour les messages de son choix et peut adapter ses requêtes en fonction des signature qu'il a déjà obtenues ;

Attaque à messages choisis (*Chosen Message Attack* (CMA)) **:** comme l'attaque précédente, l'adversaire a accès à un oracle de signature pour les messages de son choix et peut adapter ses requêtes en fonction des signatures précédentes tout en ayant, cette fois-ci, accès aux données publiques.

Pour définir la sécurité d'un schéma de signature numérique, il faut donc prendre en compte ces deux composantes qui caractérisent l'adversaire. Chaque schéma est présenté comme résistant à une attaque définie selon le couple (objectif, moyen) de l'adversaire. En utilisant nos

définitions précédentes, l'objectif appartient à l'ensemble {UBK, UF, SUF, EUF} et le moyen est compris dans {NM, KMA, SCMA, CMA}.

La sécurité standard pour les signatures est EUF-CMA : il est difficile pour l'adversaire de générer une signature sur un nouveau message, même avec accès à un oracle qui lui fournit des signatures sur des messages de son choix. Pour la construction de nos schémas, nous utiliserons plusieurs signatures classiques de la littérature : RSA, Schnorr et Boneh-Boyen dans ses deux versions.

Signature RSA. Introduite en 1978 avec le chiffrement RSA [RSA78] par RIVEST, SHAMIR et ADLEMAN, la signature RSA est aujourd'hui couramment utilisée, en particulier pour les transactions électroniques. Nous présentons ici une version plus robuste que la version initiale qui repose sur le paradigme du *hacher puis signer* [BR93].

- $\texttt{KeyGen}(1^\lambda)$ choisit aléatoirement deux grands nombres premiers p et q de longueur λ bits, une fonction de hachage cryptographique $H : \{0,1\}^* \to \mathbb{Z}_n$ et un e premier avec $\phi(n) = (p-1)(q-1)$ où ϕ est l'indicatrice d'Euler ; il calcule ensuite $d = e^{-1} \bmod \phi(n)$. La clé privée du signataire est alors d et la clé publique est (n,e) ;
- $\texttt{Sign}(m,d)$ applique la fonction de hachage sur le message m et le signe avec sa clé privée d ; la signature est égale à :

$$\sigma = H(m)^d;$$

- $\texttt{Verif}(m,\sigma,(n,e))$ s'assure que la signature est valide pour le message m par le signataire associé à la clé publique (n,e) en vérifiant l'égalité suivante :

$$H(m) = \sigma^e \bmod n;$$

  si l'égalité est vraie, il renvoie 1 et 0 sinon.

Signature de Schnorr. La signature de SCHNORR [Sch90], introduite en 1989, est particulièrement simple et très efficace. Elle est sûre sous l'hypothèse du logarithme discret DL dans le modèle de l'oracle aléatoire.

- $\texttt{Setup}(1^\lambda, 1^{\lambda'})$ génère les paramètres $pp = (p,q,g,H)$ où p,q sont premiers tels que q divise $(p-1)$, $q \geq 2^\lambda$, $p \geq 2^{\lambda'}$ et g est un élément d'ordre q dans $\mathbb{Z}_p$ et $H : \{0,1\}^* \to \mathbb{Z}_q$ est une fonction de hachage cryptographique ;
- $\texttt{KeyGen}(pp)$ choisit aléatoirement la clé privée x dans $\mathbb{Z}_q^*$ et calcule la clé publique correspondante $y = g^x \bmod p.$;
- $\texttt{Sign}(pp,m,x)$ choisit aléatoirement r dans $\mathbb{Z}_q$ et calcule $t = g^r \bmod p$, $c = H(t,m)$ et $s = r + cx \bmod q$; la signature est alors

$$\sigma = (c,s);$$

- $\texttt{Verif}(pp,m,\sigma,y)$ s'assure que la signature est valide pour le message m en calculant $t' = g^s y^{-c}$ et en vérifiant l'égalité suivante :

$$c = H(t',m).$$

Si l'égalité est vraie, il renvoie 1 et 0 sinon.

Dans le cas de la factorisation et du logarithme discret, il est recommandé d'utiliser des modules premiers d'au moins 3072 bits [ANS13].

Signature Boneh-Boyen. Introduites en 2004, les signatures Boneh-Boyen [BB04] sont les premières signatures courtes basées sur les couplages à savoir avec un nombre premier de taille au moins 256 bits. Deux variantes ont été présentées avec des propriétés de sécurité différentes. Nous décrivons ici la version la plus efficace et la plus simple des deux. Elle est prouvée EUF-SCMA sous l'hypothèse $q-\mathsf{SDH}$, dans le modèle standard.

- $\mathtt{Setup}(1^\lambda)$ retourne les paramètres $pp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ c'est-à-dire un environnement bilinéaire ;
- $\mathtt{KeyGen}(pp)$ sélectionne deux générateurs $g_1 \in \mathbb{G}_1$ et $g_2 \in \mathbb{G}_2$ ainsi qu'un nombre aléatoire γ dans $\mathbb{Z}_p$ puis calcule $Y = g_2^\gamma$ et $z = e(g_1, g_2)$; la clé privée sk est alors γ et la clé publique pk associée est (g_1, g_2, Y, z) ;
- $\mathtt{Sign}(pp, m, sk)$ calcule la signature A sur le message m avec la clé privée $sk = \gamma$:

$$A = g_1^{\frac{1}{\gamma+m}}.$$

- $\mathtt{Verif}(pp, m, A, pk)$ vérifie la signature A sur le message m, à partir de la clé publique $pk = (g_1, g_2, Y, z)$. Il teste l'égalité suivante :

$$e(A, Y \cdot g_2^m) = z.$$

Si l'égalité est vraie, il retourne 1 et 0 sinon.

Ce schéma – tout comme sa seconde version – peut se généraliser à la signature de n messages $(m_1, \ldots, m_n)$.

Signature Boneh-Boyen - sans couplage. En 2014, CANARD, COISEL, JAMBERT et TRAORÉ [CCJT13] proposent une version sans couplage de cette signature. Celle-ci est donc adaptée aux environnements où le calcul des couplages n'est pas possible telle qu'une carte SIM, par exemple. La version sans couplage de la signature de Boneh-Boyen est EUF-CMA sous l'hypothèse $q-\mathsf{SDH}$, dans le modèle de l'oracle aléatoire.

- $\mathtt{Setup}(1^\lambda)$ retourne les paramètres $pp = (p, \mathbb{G})$ où $\mathbb{G}$ est un groupe cyclique, d'ordre premier p, tel que DDH est difficile ;
- $\mathtt{KeyGen}(pp)$ sélectionne deux générateurs g et g_1 dans $\mathbb{G}$ ainsi qu'un nombre aléatoire γ dans $\mathbb{Z}_p$; la clé privée sk est alors γ et la clé publique pk associée est $Y = g^\gamma$;
- $\mathtt{Sign}(pp, m, sk)$ calcule la signature Boneh-Boyen sur le message m avec la clé privée $sk = \gamma$; étant donné l'absence de couplage, le signataire doit prouver que la signature est valide : il génère donc une preuve de connaissance [2] π que le logarithme discret de $g_1 A^{-m}$ en base A est égal au logarithme discret de Y en base g ; la signature σ est alors

2. Nous définissons en détail les preuves de connaissance et la notation utilisée à la Section 2.3.2. Ces preuves permettent à un utilisateur de prouver qu'il dispose d'un secret (ici la clé secrète sk) sans le révéler. La notation PoK$\{\alpha : \phi(\alpha)\}$ permet de noter que le secret α non révélé vérifie bien le prédicat ϕ.

constituée des deux éléments suivants :

$$A = g_1^{\frac{1}{\gamma+m}} \quad \text{et} \quad \pi = \text{PoK}\{\alpha : Y = g^{\alpha} \wedge g_1 A^{-m} = A^{\alpha}\};$$

– `Verif`(pp, m, σ, pk) vérifie la signature σ sur le message m grâce à la preuve π ; si la preuve π est valide, il retourne 1 et 0 sinon.

Première variante : signature aveugle

En 1983, CHAUM [Cha82] introduit les signatures aveugles pour les protocoles où le respect de la vie privée des utilisateurs est essentiel. Elles permettent à un receveur d'obtenir une signature sur un message, sans fournir d'information sur ce message au signataire, en envoyant une version cachée du message initial. En particulier, les signatures aveugles sont adaptées aux systèmes de paiement ou de vote électroniques. Comparées aux signatures numériques précédentes, des propriétés additionnelles sont attendues :

– il doit être impossible pour le receveur d'obtenir $L+1$ signatures après au plus L requêtes de signatures (*one-more unforgeability*) ;
– le message à signer doit rester secret pour le signataire ;
– il doit être impossible de relier entre eux deux couples (message, signature) comme émanant d'un même utilisateur ;
– il doit être impossible d'identifier pour qui a été calculé une signature aveugle – et donc de relier un couple (message, signature) à une exécution de `Sign`.

Plus formellement, la définition d'une signature aveugle est la suivante. Pour cette définition comme pour les suivantes, nous ne détaillons que les algorithmes qui diffèrent de la Définition 27 du schéma de signature numérique classique.

Définition 28 (Schéma de signature aveugle)**.** Un *schéma de signature aveugle* est constitué des quatre algorithmes (`Setup`, `KeyGen`, `BlindSign`, `Verif`). L'étape de signature aveugle est détaillée ci-après.

Signature aveugle. La signature est obtenue par un protocole interactif, noté `BlindSign`, entre un receveur $\mathcal{R}$ et un signataire $\mathcal{S}$. $\mathcal{R}$ connaît la clé publique de $\mathcal{S}$ et souhaite obtenir une signature sur le message m sans dévoiler d'information sur celui-ci. Pour cela, il masque m sous un nouveau message m'. À la réception de ce message, $\mathcal{S}$ utilise sa clé privée sk pour générer une signature σ'. À la fin du protocole, $\mathcal{R}$ reçoit σ' et en déduit la signature σ de m.

$$\sigma \leftarrow \texttt{BlindSign}(\mathcal{R}(m, pk), \mathcal{S}(sk))$$

Cependant, avec une signature aveugle, le fait que le signataire n'ait aucune information sur le message peut poser problème puisqu'il n'a aucune idée des données à signer. Dans certains contextes, il peut être intéressant d'ajouter une information supplémentaire dans cette signature.

En 1996, ABE et FUJISAKI [AF96] introduisent le concept de signature partiellement aveugle. Ici, le signataire ajoute une information, en accord avec le receveur, telle qu'une date, une période de validité ou encore le montant d'une transaction. Les propriétés attendues sont identiques à la signature aveugle.

Définition 29 (Schéma de signature partiellement aveugle)**.** Un *schéma de signature partiellement aveugle* est constitué des quatre algorithmes (`Setup`, `KeyGen`, `PartBlindSign`, `Verif`). L'étape de signature partiellement aveugle est détaillée ci-après.

Signature partiellement aveugle. La signature est obtenue par un protocole interactif, noté `PartBlindSign`, entre un receveur $\mathcal{R}$ et un signataire $\mathcal{S}$. $\mathcal{R}$ connaît la clé publique de $\mathcal{S}$ et souhaite obtenir une signature sur le message m sans dévoiler d'information sur celui-ci. De plus, $\mathcal{R}$ et $\mathcal{S}$ se mettent d'accord sur une information commune *info*, ajoutée à la signature. Cette étape peut être réalisée avant ou au cours du protocole. $\mathcal{R}$ masque m sous un message m'. À la réception de ce message, $\mathcal{S}$ utilise sa clé privée sk pour générer une signature σ' sur m' et l'information *info*. À la fin du protocole, $\mathcal{R}$ reçoit σ' et en déduit la signature σ de m et *info*.

$$\sigma \leftarrow \texttt{PartBlindSign}(\mathcal{R}(m, info, pk), \mathcal{S}(info, sk))$$

Deuxième variante : signature agrégée

En 2003, BONEH, GENTRY, LYNN et SHACHAM [BGLS03] proposent les signatures agrégées. Elles permettent de réunir n signatures, produites par n signataires différents sur n messages distincts, dans une seule et même signature. Associée aux n messages, la signature obtenue va convaincre le vérificateur que les n messages ont bien été signés par les n signataires.

Définition 30 (Schéma de signature agrégée)**.** Un *schéma de signature agrégée* est un schéma de signature (`Setup`, `KeyGen`, `Sign`, `Verif`) auquel est ajouté les deux étapes associées aux algorithmes suivants :

Agrégation. L'algorithme d'agrégation des signatures est noté `Aggregate`. Il prend en entrée les paramètres pp, un ensemble de n messages distincts $(m_1, \ldots, m_n)$ et les n signatures associées $(\sigma_1, \ldots, \sigma_n)$. Il renvoie la signature σ sur $(m_1, \ldots, m_n)$.

$$\sigma \leftarrow \texttt{Aggregate}(pp, (m_1, \ldots, m_n), (\sigma_1, \ldots, \sigma_n))$$

Vérification de l'agrégation. L'algorithme de vérification de la signature agrégée est noté `AggVerif`. Il prend en entrée les paramètres pp, la signature σ obtenue pour les messages $(m_1, \ldots, m_n)$ et les clés publiques utilisées $(pk_1, \ldots, pk_n)$. Il renvoie 1 si la signature est valide et 0 sinon.

$$\{0,1\} \leftarrow \texttt{AggVerif}(pp, (m_1, \ldots, m_n), \sigma, (pk_1, \ldots, pk_n))$$

En 2004, LYSYANSKAYA, MICALI, REYZIN et SHACHAM [LMRS04] décrivent la première signature séquentiellement agrégée. Il s'agit d'un cas spécifique où la signature obtenue est créée par chaque signataire signant l'un après l'autre c'est-à-dire en prenant en compte la signature générée auparavant, elle-même conçue à partir des signatures précédentes.

Définition 31 (Schéma de signature séquentiellement agrégée)**.** Un *schéma de signature séquentiellement agrégée* est constitué des quatre algorithmes (`Setup`, `KeyGen`, `SeqAggSign`, `Verif`). Chaque participant i détient une paire de clés (pk_i, sk_i) obtenue avec l'algorithme `KeyGen`. Les étapes de signature séquentielle et de vérification sont détaillées ci-après.

Signature séquentielle. L'algorithme de signature séquentielle est noté `SeqAggSign`. Il prend en entrée les paramètres pp, une signature agrégée σ sur un ensemble de messages distincts $(m_1, \ldots, m_{i-1})$, une clé privée sk_i et un nouveau message m_i. Il retourne la signature σ' sur $(m_1, \ldots, m_i)$.

$$\sigma' \leftarrow \texttt{SeqAggSign}(pp, \sigma, m_i, sk_i)$$

Vérification. L'algorithme de vérification de la signature séquentiellement agrégée est noté `Verif`. Il prend en entrée les paramètres pp, la signature séquentiellement agrégée σ' et les messages correspondants $(m_1, \ldots, m_n)$ ainsi que les clés publiques $(pk_1, \ldots, pk_n)$. Il renvoie 1 si la signature est valide et 0 sinon.

$$\{0,1\} \leftarrow \texttt{Verif}(pp, (m_1, \ldots, m_n), \sigma, (pk_1, \ldots, pk_n))$$

2.2.2 Codes d'authentification de messages

Les codes d'authentification de messages, notés MAC pour *Message Authentication Code*, assurent également l'intégrité des messages. Ils reposent sur la cryptographie à clé secrète c'est-à-dire que l'émetteur et le destinataire partagent une clé secrète commune. Cette clé permet ainsi de garantir l'authentification des messages puisqu'ils sont les seuls à la posséder. En contrepartie, la clé étant secrète, les MAC ne sont pas vérifiables publiquement – contrairement aux signatures numériques. Cela implique qu'il n'est pas possible d'atteindre la propriété dite de "non-répudiation" puisque toute personne capable de vérifier un MAC peut en générer un également.

Les attaques pour les MAC sont similaires à celles introduites précédemment dans le cas des signatures. En particulier, un MAC est généralement considéré comme sûr lorsqu'il est UF-CMA avec oracle de vérification, noté alors UF-CMVA.

Plus formellement, la définition d'un MAC est la suivante.

Définition 32 (Schéma de code d'authentification de message (MAC))**.** Un *schéma de code d'authentification de message* est constitué des quatre étapes associées aux algorithmes suivants :

Initialisation. L'initialisation du système est réalisée par l'algorithme `Setup`. Il prend en entrée 1^λ où λ est un paramètre de sécurité et retourne les paramètres publics du système, notés pp.

$$pp \leftarrow \texttt{Setup}(1^\lambda)$$

Génération des clés. L'algorithme de génération de clé est probabiliste et noté `KeyGen`. Il prend en entrée les paramètres pp et génère une clé secrète sk, commune à l'émetteur et au destinataire.

$$sk \leftarrow \texttt{KeyGen}(pp)$$

MAC. L'algorithme de calcul du code d'authentification de message, déterministe ou probabiliste, dénoté `MAC`. Il prend en entrée les paramètres pp, un message m et la clé secrète sk. Il retourne un code d'authentification du message m, appelé MAC de m et noté τ.

$$\tau \leftarrow \texttt{MAC}(pp, m, sk)$$

Vérification. L'algorithme de vérification est déterministe et noté `Verif`. Il prend en entrée les paramètres *pp*, un message *m* donné, un MAC τ et la clé secrète *sk*. Si le MAC τ est valide pour *m* et *sk*, il retourne 1 et 0 sinon.

$$\{0,1\} \leftarrow \texttt{Verif}(pp, m, \tau, sk)$$

MAC algébrique ($\mathsf{MAC}_{\mathsf{GGM}}$). En 2014, CHASE, MEIKLEJOHN et ZAVERUCHA [CMZ14] introduisent deux schémas de MACs algébriques. Contrairement aux MACs habituels, ceux-ci sont construits en utilisant les opérations de groupe cyclique. Les deux schémas qu'ils proposent ont été prouvés UF-CMVA sous l'hypothèse DDH. En particulier, nous utiliserons dans ce travail le schéma noté $\mathsf{MAC}_{\mathsf{GGM}}$ et détaillé ci-après pour n messages distincts. Il s'agit en fait d'une généralisation du schéma [DKPW12] de DODIS, KILTZ, PIETRZAK et WICHS. La particularité de leur schéma est que des paramètres publics peuvent être générés. Cette propriété nous sera utile dans le cas d'accréditations anonymes avec vérification par clé (voir Chapitre 4).

- $\texttt{Setup}(1^\lambda)$ retourne les paramètres publics $pp = (p, \mathbb{G}, g, h)$ où $\mathbb{G}$ est un groupe cyclique, d'ordre premier p, et g, h sont deux générateurs aléatoires de $\mathbb{G}$ tel que $\log_g h$ soit inconnu ;
- $\texttt{KeyGen}(pp)$ génère la clé secrète *sk* égale à $\vec{x} = (x_0, \ldots, x_n)$ où chaque x_i est un élément aléatoire de $\mathbb{Z}_p$ et choisit une valeur aléatoire $\tilde{x}_0$ dans $\mathbb{Z}_p$ pour construire l'engagement $C_{x_0} = g^{x_0} h^{\tilde{x}_0}$ sur le secret x_0 (voir Définition 33) ; des paramètres publics *iparams* peuvent être également définis pour l'émetteur tels que $iparams = (C_{x_0}, X_1 = h^{x_1}, \ldots, X_n = h^{x_n})$;
- $\texttt{MAC}(pp, \vec{m}, sk)$ produit le MAC $\tau = (u, u')$ sur le message $\vec{m} = (m_1, \ldots, m_n)$ à partir de la clé secrète $\vec{x}$:
$$\tau = (u, u' = u^{x_0 + x_1 m_1 + \cdots + x_n m_n})$$
avec u aléatoire dans $\mathbb{G}$ et différent de 1 ;
- $\texttt{Verif}(pp, \vec{m}, \tau, sk)$ teste la validité du MAC par rapport au message $\vec{m}$ en vérifiant les propositions suivantes :
$$u \neq 1 \quad \text{et} \quad u' = u^{x_0 + x_1 m_1 + \cdots + x_n m_n}.$$
Si elles sont vérifiées, il retourne 1 et 0 sinon.

2.3 Autres primitives

Des primitives cryptographiques additionnelles permettent de répondre à des problématiques plus spécifiques. Ainsi, nous présentons ici la notion de mise en gage et de preuves à divulgation nulle de connaissance. Enfin, nous introduirons un modèle de protection de la vie privée connue sous le nom de confidentialité différentielle.

2.3.1 Mise en gage

Une mise en gage – ou engagement – permet à un prouveur de mettre en gage une valeur auprès d'un vérificateur sans lui révéler immédiatement. Une fois engagée, il n'est pas possible

de la modifier mais elle pourra être révélée plus tard si besoin. De cette façon, un vérificateur a l'assurance que la valeur mise en gage est bien celle choisie initialement puisque le prouveur ne peut plus en changer. Cette notion a été formalisée pour la première fois en 1988 par BRASSARD, CHAUM et CRÉPEAU [BCC88].

Les propriétés attendues d'un schéma de mise en gage sont les suivantes :

- il doit être impossible d'obtenir de l'information sur la valeur à partir d'une mise en gage, seule le moment où elle est révélée permet de connaître sa valeur ;
- une fois mise en gage, une valeur ne peut pas être modifiée.

Définition 33 (Schéma de mise en gage)**.** Un *schéma de mise en gage* est constitué des trois étapes associées aux algorithmes suivants :

Initialisation. L'initialisation du système est réalisée par l'algorithme `Setup`. Il prend en entrée 1^λ où λ est un paramètre de sécurité et retourne les paramètres publics du système, notés pp.

$$pp \leftarrow \texttt{Setup}(1^\lambda)$$

Mise en gage. L'algorithme de mise en gage est probabiliste et noté `Commit`. Il prend en entrée les paramètres pp et un message m à mettre en gage. Il génère un engagement C du message m et retourne également l'information auxiliaire R nécessaire pour l'ouverture.

$$(C,R) \leftarrow \texttt{Commit}(pp,m)$$

Vérification. L'algorithme de vérification de l'engagement est déterministe et est noté `Open`. Il prend en entrée les paramètres pp, le message m associé au couple (C,R) contenant la mise en gage et l'information R. Il retourne 1 si la mise en gage est valide et 0 sinon.

$$\{0,1\} \leftarrow \texttt{Open}(pp,(C,R),m)$$

De façon plus formelle, les propriétés de sécurité deviennent :

Indistinguabilité (*hiding*) **:** un ensemble de mises en gage sur un message m doit être – parfaitement ou calculatoirement – indistinguable d'un ensemble de mises en gage sur un message m' différent de m ;

Résistance aux collisions (*binding*) **:** pour tout message m et toute mise en gage (C,R), il doit être – parfaitement ou calculatoirement – impossible de trouver un nouveau message m' tel que $\texttt{Open}(pp,(C,R),m) = \texttt{Open}(pp,(C,R'),m') = 1$.

Il est à noter, cependant, qu'une mise en gage ne peut pas être à la fois parfaitement indistinguable et parfaitement résistante aux collisions [Dam99].

Engagement de Pedersen. Introduit en 1992, le schéma de mise en gage le plus utilisé en protection de la vie privée est celui de PEDERSEN [Ped92]. Ce schéma de mise en gage est parfaitement indistinguable et calculatoirement résistant aux collisions, sous l'hypothèse du logarithme discret DL.

- $\texttt{Setup}(1^\lambda)$ retourne les paramètres publics $pp = (p,\mathbb{G},g,h)$ où $\mathbb{G}$ est un groupe cyclique d'ordre p et g,h sont deux générateurs aléatoires de $\mathbb{G}$;

– `Commit`(pp, m) génère la mise en gage C sur le message $m \in \mathbb{Z}_p$ tel que

$$C = g^m h^r$$

avec r aléatoire dans $\mathbb{Z}_p$ et il retourne alors (C, r) ;
– `Open`$(pp, m, (C, R))$ vérifie la validité de la mise en gage par rapport au message m en vérifiant l'égalité suivante : $C = g^m h^r$.

2.3.2 Preuves de connaissance à divulgation nulle de connaissance

Dès 1985, GOLDWASSER, MICALI et RACKOFF [GMR85, GMR89] montrent qu'il est possible de prouver l'appartenance d'un mot à un langage, sans révéler ce mot. En 1987, FEIGE, FIAT et SHAMIR [FFS87, FFS88] introduisent la notion de preuve de connaissance à divulgation nulle de connaissance. Par abus de langage et pour fin de simplification, nous les appelons simplement preuve à divulgation nulle de connaissance. Elles permettent à un utilisateur de prouver à un tiers qu'il connaît un secret vérifiant un prédicat particulier, sans révéler d'information sur le secret en question. En pratique, ce protocole permet de prouver la connaissance d'un témoin pour l'élément secret.

Les preuves de connaissance peuvent être vues comme une généralisation des schémas d'authentification. En effet, le secret détenu par l'utilisateur peut très bien être sa clé privée qui permet, dans de nombreux systèmes, de l'identifier. Ces preuves sont désormais fondamentales en cryptographie moderne. Plus formellement, les définitions de preuves de connaissance et de preuve à divulgation nulle de connaissance sont les suivantes.

Définition 34 (Preuve de connaissance (PK))**.** Une *preuve de connaissance*, notée PK (*Proof of Knowledge*), est un protocole interactif entre un prouveur et un vérificateur qui permet au vérificateur de s'assurer que le prouveur connaît un secret attestant la véracité d'un prédicat donné. Le protocole doit satisfaire les propriétés suivantes :

Complétude (*Completeness*) **:** une preuve générée par un prouveur honnête – qui connaît le secret – sera acceptée par un vérificateur honnête avec une probabilité écrasante ;

Validité (*Soundness*) **:** une preuve générée par une prouveur malhonnête – qui ne connaît pas le secret – sera refusée par un vérificateur honnête sauf avec une probabilité négligeable.

Pour prouver la validité d'une preuve de connaissance, il suffit généralement de démontrer l'existence d'un extracteur en temps polynômial qui interagit avec tout prouveur et est capable de retourner un témoin pour le secret considéré dont il n'a pas connaissance. Nous en donnerons un exemple par la suite.

Définition 35 (Preuve à divulgation nulle de connaissance (ZKPK))**.** Une preuve de connaissance est appelée *preuve à divulgation nulle de connaissance*, notée ZKPK (pour *Zero-Knowledge*), si le vérificateur n'apprend aucune information additionnelle à part la certitude que le prouveur connaît bien le secret. La preuve à divulgation nulle de connaissance vérifie alors la propriété additionnelle suivante :

Non-divulgation (*Zero-Knowledge*) **:** il est possible de construire un simulateur dont la sortie est indistinguable d'une instance réelle de la preuve exécutée entre le vérificateur et le prouveur.

Pour satisfaire pleinement la propriété de divulgation nulle, il est nécessaire d'avoir au minimum quatre échanges [GK96] entre les participants. Dans notre cas, nous ne considérons que des ZKPK où cette propriété est satisfaite uniquement face à un vérificateur honnête puisqu'alors trois passes suffisent. Elles sont également appelées Σ-protocoles où le symbole Σ schématise alors les échanges.

En outre, afin de prouver la non-divulgation d'une preuve de connaissance, il faut prendre en compte tous les éléments échangés entre le prouveur et tout vérificateur. De fait, si l'ensemble de ces éléments peuvent être simulés sans la connaissance de ce secret alors cela signifie que la preuve ne révèle pas d'information. Contrairement au prouveur, le simulateur peut faire plusieurs essais avant d'obtenir une simulation correcte. Ainsi, un simulateur peut revenir en arrière autant de fois qu'il le souhaite selon les réponses fournies par le vérificateur – appelé parfois *rembobinage*.

Par la suite, pour représenter les preuves à divulgation nulle de connaissance, nous utilisons la notation usuelle introduite par CAMENISCH et STADLER [CS97] :

$$\text{PoK}\{\alpha, \beta, \cdots : \text{ prédicat sur } \alpha, \beta, \cdots\}$$

où les lettres Grecques correspondent aux secrets connus par le prouveur.

Nous présentons ici les preuves utilisées dans la suite du mémoire. Pour plus de détails, nous invitons le lecteur à se référer à [Cam98].

Protocole d'identification interactif de Schnorr. En 1990, SCHNORR propose un protocole qui est aujourd'hui à la base de nombreuses constructions. Il permet à un prouveur de prouver la connaissance du logarithme discret en base g d'une valeur publique $y = g^x$ connue des deux participants et repose donc sur le problème du logarithme discret DL. Par la suite, nous notons cette preuve de la manière suivante :

$$\text{PoK}\{\alpha : y = g^\alpha\}.$$

Le *protocole d'identification de Schnorr* fonctionne comme suit.

- $\texttt{Setup}(1^\lambda)$ retourne les paramètres publics $pp = (p, q, g)$ où p et q sont deux nombres premiers tels que q divise $p-1$, g est élément de $\mathbb{Z}_p^*$ d'ordre q ;
- $\texttt{KeyGen}(pp)$ génère la clé publique $y = g^x \bmod p$ associée la clé secrète $x \in \mathbb{Z}_q^*$; elle est utilisée par le prouveur pour prouver sa connaissance du secret x ;
- $\text{PK}(pp, \mathcal{P}(x), \mathcal{V}(y))$ est un protocole interactif entre le prouveur et le vérificateur, composé des trois étapes suivantes :

  Engagement : le prouveur choisit aléatoirement a dans $\mathbb{Z}_q^*$ et génère ensuite le témoin $t = g^a \bmod p$ qu'il envoie au vérificateur ;

  Challenge : ensuite, le vérificateur choisit aléatoirement c dans $\mathbb{Z}_q$ et envoie ce challenge c au prouveur ;

  Réponse : le prouveur calcule la réponse $s = a + cx \bmod q$ et l'envoie au vérificateur.

  Si l'égalité $t = g^s y^{-c}$ est vérifiée alors le vérificateur est convaincu que le prouveur connaît bien le secret associé à y.

Ce protocole est bien une preuve à divulgation nulle de connaissance puisqu'il satisfait les propriétés de complétude, de validité et de divulgation nulle comme montré ci-après.

Complétude. Par simple vérification, si $y = g^x \bmod p$ alors nous avons

$$g^s y^{-c} = g^{a+cx} g^{-cx} = t \bmod p.$$

Validité. Cette propriété repose sur la propriété d'extraction de la preuve de connaissance impliquée par l'existence d'un extracteur. En effet, si un prouveur est capable de répondre à deux challenges distincts c et c', pour un même témoin t, alors il est possible de retrouver le secret x. En effet, à partir des transcriptions (t, c, s) et (t, c', s') avec $c \neq c'$, l'extracteur peut calculer le secret comme suit :

$$\frac{s-s'}{c-c'} = \frac{a+cx-a-c'x}{c-c'} = \frac{(c-c')x}{c-c'} = x \bmod q.$$

Non-divulgation (vérificateur honnête). Pour un vérificateur honnête – c'est-à-dire qu'il suit normalement les étapes du protocole, la transcription obtenue est (t, c, s) où c est aléatoire et indépendant. La propriété de divulgation nulle repose alors sur l'existence d'un simulateur Sim, en temps polynomial, qui fonctionne comme suit :

1. Sim choisit aléatoirement c et s dans $\mathbb{Z}_q$ puis calcule $t = g^s h^{-c} \bmod p$;
2. Sim retourne (t, c, s).

Le triplet (t, c, s) est alors indistinguable d'une réelle exécution.

En outre, ce protocole peut être utilisé pour fournir des preuves plus complexes. Tout d'abord, il peut être généralisé pour prouver la connaissance d'une représentation d'un élément public y dans la base $(g_1, \ldots, g_n)$. Le but du prouveur est alors de prouver la connaissance du secret qui est un n—uplet $(x_1, \ldots, x_n)$ tel que $y = g_1^{x_1} \ldots g_n^{x_n}$ où chaque x_i est dans $\mathbb{Z}_q^*$. Pour cela, le prouveur génère n mises en gage $t_i = g_i^{a_i}$, avec a_i aléatoire, pour chaque composante $\{x_i\}_1^n$ du secret. De même, en réponse au challenge c du vérificateur, il construit n réponses $s_i = a_i + cx_i \bmod q$. Nous notons cette preuve comme suit :

$$\text{PoK}\{\alpha_1, \ldots, \alpha_n : y = g^{\alpha_1} \ldots g^{\alpha_n}\}.$$

Preuve d'égalité de logarithmes discrets. Le protocole de Schnorr peut également être généralisé pour prouver l'égalité de logarithmes discrets dans des bases différentes. Cette preuve sera notée de la manière suivante :

$$\text{PoK}\{\alpha : y_1 = g_1^{\alpha} \wedge y_2 = g_2^{\alpha}\}.$$

Une *preuve d'égalité de logarithmes discrets* fonctionne comme suit.

- $\texttt{Setup}(1^\lambda)$ retourne les paramètres publics $pp = (p, q, g_1, g_2)$ où p et q sont deux nombres premiers tels que q divise $p-1$ et g_1, g_2 sont deux éléments de $\mathbb{Z}_p^*$ d'ordre q ;
- $\texttt{KeyGen}(pp)$ génère les valeurs publiques $y_1 = g_1^x \bmod p$ et $y_2 = g_2^x \bmod p$ associées au secret x dans $\mathbb{Z}_q^*$;
- $\texttt{PK}(pp, \mathcal{P}(x), \mathcal{V}(y))$ est un protocole interactif entre le prouveur et le vérificateur, composé des trois étapes suivantes :

Engagement : le prouveur choisit aléatoirement a dans $\mathbb{Z}_q^*$ et génère les deux témoins $t_1 = g_1^a \bmod p$ et $t_2 = g_2^a \bmod p$ qu'il envoie au vérificateur ;

Challenge : ensuite, le vérificateur choisit aléatoirement c dans $\mathbb{Z}_q$ et envoie ce challenge c au prouveur ;

Réponse : le prouveur calcule la réponse $s = a + cx \bmod q$ et l'envoie au vérificateur.

Si les égalités $t_1 = g_1^s y^{-c}$ et $t_2 = g_2^s y^{-c}$ sont vérifiées alors le vérificateur est convaincu que les logarithmes discrets connus du prouveur sont égaux.

Preuve du "ou logique". La généralisation suivante du protocole de Schnorr permet de prouver la connaissance d'une valeur – ou plusieurs parmi n pour n fixé – dans un ensemble, sans révéler de laquelle il s'agit. Nous la présentons ici dans le cas simple d'un ensemble de deux éléments dont un est connu par le prouveur. Nous noterons cette preuve de la manière suivante :

$$\text{PoK}\{\alpha : y_1 = g^\alpha \vee y_2 = g^\alpha\}.$$

Une *preuve à divulgation nulle de connaissance d'un élément parmi deux* fonctionne comme suit :

- $\texttt{Setup}(1^\lambda)$ retourne les paramètres publics $pp = (p, q, g)$ où p et q sont deux nombres premiers tels que q divise $p - 1$ et g est un élément de $\mathbb{Z}_p^*$ d'ordre q ;
- $\texttt{KeyGen}(pp)$ génère les valeurs publiques $y_1 = g^x \bmod p$ où x est le secret du prouveur dans $\mathbb{Z}_q^*$ et y_2 est un élément aléatoire dans le groupe engendré par g ;
- $\text{PK}(pp, \mathcal{P}(x), \mathcal{V}(y))$ est un protocole interactif entre le prouveur et le vérificateur, composé des trois étapes suivantes :

  Engagement : le prouveur choisit aléatoirement a dans $\mathbb{Z}_q^*$ puis génère le témoin $t_1 = g^a \bmod p$; pour la valeur inconnue, il simule un témoin en calculant $t_2 = g^{s'} y^{-c'}$ où s', c' sont tirés aléatoirement dans $\mathbb{Z}_q$; les deux témoins sont envoyés au vérificateur ;

  Challenge : ensuite, le vérificateur choisit aléatoirement c dans $\mathbb{Z}_q$ et envoie ce challenge c au prouveur ;

  Réponse : le prouveur calcule les deux valeurs $C = c \oplus c'$ et $s = a + Cx \bmod q$ et envoie (C, c', s, s') au vérificateur.

  Le vérificateur est convaincu que le prouveur connaît l'une des deux valeurs sans savoir laquelle si toutes les égalités $c = C \oplus c'$, $t_1 = g^s y^{-c}$ et $t_2 = g^{s'} y^{-c'}$ sont vérifiées.

Les preuves présentées jusqu'ici nécessitent des interactions, c'est-à-dire des échanges de messages, entre le prouveur et le vérificateur. Cependant, il existe aussi des versions non-interactives des preuves à divulgation nulle de connaissance qui permettent d'éviter ces échanges.

Heuristique de Fiat-Shamir.

En 1986, FIAT et SHAMIR [FS87] proposent une méthode pour convertir les preuves à divulgation nulle de connaissance en trois passes face à un vérificateur honnête en protocoles non-interactifs, à divulgation nulle. L'idée est de ne plus laisser le vérificateur calculer seul le challenge mais de le construire de telle sorte que le vérificateur et le prouveur puisse chacun le calculer de leur côté. En pratique, cela est effectué avec une fonction de hachage qui prend en entrée la mise en gage du protocole et les valeurs publiques associées. Les preuves obtenues

sont alors notées NIZKPK (pour *Non-Interactive*). Très efficace, cette méthode est couramment adoptée. Pour l'illustrer, nous l'appliquons ici au protocole de Schnorr présenté auparavant.

Dans sa version non-interactive, le protocole de Schnorr devient :

- $\texttt{Setup}(1^\lambda)$ retourne les paramètres publics $pp = (p, q, g, H)$ où p et q sont deux nombres premiers tels que q divise $p-1$, g est élément de $\mathbb{Z}_p^*$ d'ordre q et $H : \{0,1\}^* \to \mathbb{Z}_q$ est une fonction de hachage ;
- $\texttt{KeyGen}(pp)$ génère $y = g^x \bmod p$ associé au secret $x \in \mathbb{Z}_q^*$;
- $\texttt{PK}(pp, x)$ est exécuté par le prouveur et comprend les trois étapes suivantes :

  Engagement : le prouveur choisit aléatoirement a dans $\mathbb{Z}_q^*$ et génère le témoin $t = g^a \bmod p$;

  Challenge : le prouveur génère $c = H(t||g||y)$ qui fait office de challenge ;

  Réponse : le prouveur calcule la réponse $s = a + cx \bmod q$ et envoie au vérificateur le couple (c, s).

  Le vérificateur calcule alors $t' = g^s y^{-c}$ et est convaincu que le prouveur connaît bien le secret x associé à y si l'égalité $c = H(t'||g||y)$ est vérifiée.

Dans la suite de ce manuscrit, nous utiliserons la même notation que la preuve à divulgation nulle sous-jacente soit interactive ou non :

$$\text{PoK}\{\alpha_1, \ldots, \alpha_n : \text{prédicat}(\alpha_1, \ldots, \alpha_n)\}.$$

À noter qu'en 2008, GROTH et SAHAI [GS08] proposent un nouveau système de preuve non-interactives qui est à la fois efficace et prouvé sûr dans le modèle standard. En particulier, celui-ci est adapté aux environnements bilinéaires, c'est pourquoi nous ne le présentons pas davantage dans ce mémoire.

Vérificateur désigné (DVP). En 1996, JAKOBSSON, SAKO et IMPAGLIAZZO [JSI96] introduisent la notion de preuve pour un vérificateur désigné, notée DVP (*Designated Verifier Proof*). Il s'agit d'un type particulier de preuve de connaissance à divulgation nulle de connaissance générée pour un vérificateur choisi au préalable. En pratique, l'idée est de construire une preuve du "ou" qui est liée d'une part à la clé publique du vérificateur et d'autre part au prédicat souhaité. Par conséquent, étant donné sa connaissance de la clé secrète, le vérificateur est le seul à pouvoir vérifier le prédicat et donc être convaincu par la preuve. Pour tous les autres, obtenir cette preuve est inutile puisqu'elle ne donne aucune information : le vérificateur est toujours en mesure de la générer à partir de sa clé secrète.

2.3.3 Confidentialité différentielle

L'anonymisation de données est une thématique complémentaire à la cryptographie. Elle a pour objectif principal de supprimer tout lien existant entre des données et les individus qu'elles concernent.

En 2006, DWORK, MCSHERRY, NISSIM et SMITH [DMNS06, Dwo06] introduisent un nouveau modèle de respect de la vie privée pour lequel ils ont reçu le prix Gödel en 2017, appelé *confidentialité différentielle* et noté DP (*Differential Privacy*). Contrairement aux modèles plus anciens comme la k-anonymité [Swe98, Swe02] et ses dérivés [MKGV07, NAC07, LLV07], la

confidentialité différentielle permet de limiter l'information révélée sur un individu particulier de l'ensemble de données considéré.

L'idée principale de la confidentialité différentielle est que la présence ou l'absence d'un individu dans une base de données ne doit pas affecter le résultat obtenu par le processus d'anonymisation. Cette notion est propre à la requête effectuée ou à l'algorithme et non pas à la base de données sur laquelle ils sont appliqués. Pour cela, la confidentialité différentielle apporte de l'aléatoire aux données initiales. Un algorithme en charge d'ajouter le bruit sur la sortie est appelé un *mécanisme*. En outre, deux bases de données D_1 et D_2 sont dites *voisines* si elles diffèrent d'au plus un enregistrement – que ce soit par ajout ou suppression d'une entrée.

De façon formelle, la définition de la confidentialité différentielle est la suivante.

Définition 36 (Confidentialité différentielle (ε-DP))**.** Soit un mécanisme aléatoire $\mathcal{M} : \mathcal{D} \to \mathbb{R}$. Pour ε un réel positif, $\mathcal{M}$ satisfait la confidentialité différentielle pour le paramètre de vie privée ε si, pour toutes bases de données voisines D_1 et D_2 et pour toute sortie R produite par le mécanisme, l'équation suivante est vérifiée :

$$\mathbb{P}(\mathcal{M}(D_1) \in R) \leq \exp(\varepsilon) \times \mathbb{P}(\mathcal{M}(D_2) \in R).$$

En d'autres termes, la confidentialité différentielle pour le paramètre ε garantit que, pour toute paire de bases voisines, la présence ou l'absence d'un individu n'affecte le résultat du mécanisme qu'au plus par un facteur $\exp(\varepsilon)$ sur la probabilité de la sortie de celui-ci. La confidentialité différentielle pour les paramètres de vie privée ε et δ [DKM$^+$06] est une version plus souple de de ce modèle où, pour δ un réel positif, l'équation devient

$$\mathbb{P}(\mathcal{M}(D_1) \in R) \leq \exp(\varepsilon) \times \mathbb{P}(\mathcal{M}(D_2) \in R) + \delta.$$

Si la confidentialité différentielle est appliquée à plusieurs reprises sur une même base, la quantité ε – et donc l'information révélée – augmente en conséquence. En effet, un théorème de composition séquentielle [McS09] a été prouvé pour ce modèle. Ainsi, chaque nouvelle requête représente un coût supplémentaire, ce qui amène parfois à parler de *budget* de respect de la vie privée.

Théorème 2 (Composition Séquentielle)**.** *L'application séquentielle de mécanismes $\{\mathcal{M}_i\}$, où chacun assure la confidentialité différentielle pour ε_i, satisfait la confidentialité différentielle pour $\left(\sum_i \varepsilon_i\right)$.*

La quantité de bruit à ajouter est calculée selon la notion de *sensibilité globale*. Pour une requête donnée, celle-ci mesure la différence maximale possible en sortie pour deux bases voisines. Plus cette différence est petite, plus le bruit ajouté pourra être faible.

Définition 37 (Sensibilité)**.** La *sensibilité* Δ d'une fonction $f : \mathcal{D} \to \mathbb{R}^n$ est définie comme suit :

$$\Delta f = \max_{\substack{D_1, D_2 \in \mathcal{D} \\ \text{voisines}}} ||f(D_1) - f(D_2)||.$$

Ainsi définie, la sensibilité permet de paramétrer les mécanismes utilisés pour fournir un bruit adéquat. Un des mécanismes le plus couramment utilisé est celui de Laplace. Le bruit

ajouté est ici généré en fonction de la distribution de Laplace qui a pour densité de probabilité $f(x) = \frac{1}{2\lambda} \exp(-\frac{|x|}{\lambda})$.

Théorème 3 (Mécanisme de Laplace)**.** *Pour toute fonction $f : \mathcal{D} \rightarrow \mathbb{R}^n$, le mécanisme $\mathcal{M}$ tel que*

$$\mathcal{M}(\mathcal{D}) = f(\mathcal{D}) + < \mathcal{L}_1(\lambda) + \cdots + \mathcal{L}_d(\lambda) >$$

satisfait la confidentialité différentielle pour le paramètre ε si les $\mathcal{L}_i$ sont des variables de Laplace indépendantes et identiquement distribuées avec $\lambda = \frac{\Delta f}{\varepsilon}$ où Δf est la sensibilité de f.

Conclusion

Dans ce second chapitre, nous avons défini l'ensemble des primitives cryptographiques à la base de nos contributions. Ainsi, nous avons abordé le cas de la confidentialité des données avec le chiffrement à clé publique. Ensuite, nous avons étudié les problématiques d'authentification et d'intégrité des échanges à travers les signatures numériques et les codes d'authentification de messages. Pour finir, nous avons parcouru des primitives particulières permettant de répondre à des besoins précis à savoir les mises en gage, les preuves de connaissance et la confidentialité différentielle.

Au cours des prochains chapitres, nous allons étudier les différentes contributions de la thèse. En particulier, le premier axe de travail concerne les accréditations anonymes.

Deuxième partie

Accréditations anonymes avec vérification par clé

Chapitre 3

Nouvelles primitives

Dans ce chapitre, nous introduisons trois nouvelles constructions. Nous décrivons un MAC algébrique [BBDT16], un MAC séquentiellement agrégé [ABBT16] ainsi qu'une signature partiellement aveugle [BBD+17]. Nous définissons les propriétés attendues de chaque nouveau schéma et réalisons les preuves de sécurité.

Sommaire

3.1 Notre MAC algébrique amélioré

À la Section 2.2.2, nous avons présenté les MACs algébriques, introduits en 2014 par CHASE, MEIKLEJOHN et ZAVERUCHA [CMZ14]. Cependant, les schémas proposés nécessitent d'avoir autant de clés secrètes que de messages. Nous présentons ici un nouveau schéma de MAC algébrique basé sur la variante sans couplage du schéma de signature Boneh-Boyen, présentée à la Section 2.2.1. La particularité de notre schéma est qu'une seule clé secrète suffit, quelque soit le nombre de messages considérés. En outre, cette construction est adaptable pour permettre une vérification publique par tous et ce sans connaissance de la clé secrète.

Nous présentons le modèle de sécurité d'un MAC algébrique. Nous détaillons ensuite notre nouveau schéma de MAC pour un message puis sa généralisation à un nombre fini de messages. Pour finir, nous réalisons les preuves de sécurité de ces deux versions. Ces constructions ont été publiées à SAC 2016, dans l'article intitulé *"Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials"* [BBDT16].

3.1.1 Description

Modèle de sécurité

La propriété attendue d'un schéma de MAC algébrique (`Setup`, `KeyGen`, `MAC`, `Verif`) est la *résistance aux falsifications universelles contre des attaques adaptatives à messages choisis avec oracle de vérification*, notée UF-CMVA. Autrement dit, même en ayant accès à deux oracles, l'un qui génère un MAC et l'autre qui teste la validité, il doit être impossible de produire un nouveau MAC valide τ sur un message m qui n'a jamais été utilisé lors d'une requête à l'oracle de génération de MAC.

Une version plus forte de cette propriété, notée sUF-CMVA – pour *strong* UF-CMVA, autorise le cas où m a été envoyé à l'oracle de génération de MAC sous réserve que le nouveau MAC τ' soit différent de ceux générés par cet oracle. Plus formellement, la propriété sUF-CMVA est définie par le jeu suivant entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$:

- **Initialisation :** le challenger $\mathcal{C}$ exécute `Setup` et `KeyGen` et obtient les paramètres publics pp et la clé secrète sk, envoyés ensuite à $\mathcal{A}$;
- **Requêtes de MAC :** $\mathcal{A}$ effectue des requêtes à un oracle $\mathcal{O}_{\texttt{MAC}}$ pour obtenir un MAC τ associé au message m choisi ;
- **Requêtes de vérification :** $\mathcal{A}$ effectue des requêtes de vérification à un oracle $\mathcal{O}_{\texttt{Verif}}$ pour déterminer si une paire (τ, m) correspond à un MAC valide τ sur m ;
- **Sortie :** après au plus q requêtes à l'oracle de MAC et q_v requêtes à l'oracle de vérification, $\mathcal{A}$ retourne sa contrefaçon (τ', m') ; il remporte le jeu si les conditions suivantes sont vérifiées : (1) τ' est un MAC valide pour m' c'est-à-dire que $\texttt{Verif}(pp, sk, \tau', m')$ retourne 1 et (2) τ n'a pas été généré par l'oracle de génération de MAC au cours du jeu.

La Figure 3.1 représente la propriété sous forme d'expérience de sécurité, notée $\texttt{Exp}^{\text{sUF-CMVA}}_{\mathcal{A}}$.

$\texttt{Exp}^{\text{sUF-CMVA}}_{\mathcal{A}}(1^\lambda)$
1. $pp \leftarrow \texttt{Setup}(1^\lambda)$;
2. $sk \leftarrow \texttt{KeyGen}(pp)$;
3. $(\tau, m') \leftarrow \mathcal{A}^{\mathcal{O}_{\texttt{MAC}}, \mathcal{O}_{\texttt{Verif}}}(pp)$;
4. Si $(\tau' m,')$ a été généré par $\mathcal{O}_{\texttt{MAC}}$ alors retourner 0 ;
5. Retourner $\texttt{Verif}(pp, sk, m', \tau')$.

FIGURE 3.1 – Résistance aux falsifications

Présentation du schéma $\mathsf{MAC}_{\mathsf{BB}}$

Notre schéma de MAC algébrique pour un seul message est noté $\mathsf{MAC}_{\mathsf{BB}}$. Il est défini par les quatre algorithmes suivants :

- $\texttt{Setup}(1^\lambda)$ retourne les paramètres publics du système $pp = (p, \mathbb{G}, g, h, g_0, g_1)$ où $\mathbb{G}$ est un groupe cyclique d'ordre premier p, de longueur λ bits, et g, h, g_0, g_1 sont quatre générateurs aléatoires de $\mathbb{G}$;
- $\texttt{KeyGen}(pp)$ génère la clé secrète y de l'émetteur, aléatoire dans $\mathbb{Z}_p^*$ et calcule, si besoin, une valeur publique correspondante $Y = g_0^y$;

- $\texttt{MAC}(pp, m, y)$ calcule $A = (g_1^m g^s h)^{\frac{1}{y+r}}$ avec r et s aléatoires dans $\mathbb{Z}_p$; le MAC du message m correspond alors au triplet (A, r, s) ;
- $\texttt{Verif}(pp, m, (A, r, s), y)$ teste la validité du MAC (A, r, s) par rapport au message m en vérifiant l'équation suivante :

$$(g_1^m g^s h)^{\frac{1}{y+r}} = A.$$

Théorème 4. *Notre schéma de MAC algébrique* $\mathsf{MAC}_{\mathsf{BB}}$ *est* sUF-CMVA *sous l'hypothèse* $q-\mathsf{SDH}-\mathsf{III}$.

Démonstration. La preuve du Théorème 4 est disponible à la Section 3.1.2. □

Généralisation à n messages : schéma $\mathsf{MAC}^n_{\mathsf{BB}}$

Le schéma $\mathsf{MAC}_{\mathsf{BB}}$ peut être généralisé à des vecteurs de n messages $\vec{m} = (m_1, \ldots, m_n)$. Cette généralisation est notée $\mathsf{MAC}^n_{\mathsf{BB}}$ et est définie par les quatre algorithmes suivants :

- $\texttt{Setup}(1^\lambda)$ retourne les paramètres publics du système $pp = (p, \mathbb{G}, g, h, g_0, g_1, \ldots, g_n)$ où $\mathbb{G}$ est un groupe cyclique d'ordre premier p, de longueur λ bits, et $g, h, g_0, g_1, \ldots, g_n$ sont des générateurs aléatoires de $\mathbb{G}$;
- $\texttt{KeyGen}(pp)$ génère la clé secrète y de l'émetteur, aléatoire dans $\mathbb{Z}_p^*$ et calcule, si besoin, une valeur publique correspondante $Y = g_0^y$;
- $\texttt{MAC}(pp, \vec{m}, y)$ calcule $A = (g_1^m g_2^{m_2} \ldots g_n^{m_n} g^s h)^{\frac{1}{y+r}}$, avec r et s aléatoires dans $\mathbb{Z}_p$, sur le vecteur de message $\vec{m} = (m_1, \ldots, m_n)$; le MAC de $\vec{m}$ correspond alors au triplet (A, r, s) ;
- $\texttt{Verif}(pp, \vec{m}, (A, r, s), y)$ teste la validité du MAC (A, r, s) par rapport aux messages $\vec{m} = (m_1, \ldots, m_n)$ en vérifiant l'équation suivante :

$$(g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h)^{\frac{1}{y+r}} = A.$$

Vérification publique. Ce schéma peut être adapté pour permettre des vérifications publiques : un destinataire n'a alors pas besoin de connaître la clé secrète pour utiliser l'algorithme de vérification $\texttt{Verif}$. En effet, pour un MAC (A, r, s) sur un vecteur de messages $\vec{m} = (m_1, \ldots, m_n)$, nous avons les relations suivantes :

$$\begin{aligned} & A && = (g_1^{m_1} \ldots g_n^{m_n} g^s h)^{\frac{1}{y+r}} \\ \Rightarrow\ & A^{y+r} && = g_1^{m_1} \ldots g_n^{m_n} g^s h \\ \Rightarrow\ & A^y && = g_1^{m_1} \ldots g_n^{m_n} g^s h \cdot A^{-r} \\ \Rightarrow\ & A^y && = B \end{aligned}$$

Ainsi, l'émetteur du MAC a juste à fournir son MAC (A, r, s) associé à $\vec{m}$ et une preuve de connaissance π définie comme suit :

$$\pi = \mathrm{PoK}\{\gamma : B = A^\gamma \wedge Y = g_0^\gamma\}.$$

Tout destinataire, y compris celui n'ayant pas la connaissance de la clé secrète, est convaincu que la MAC est valide à partir de (A, r, s) et π.

Théorème 5. *Notre schéma de MAC algébrique* $\mathsf{MAC}^n_{\mathsf{BB}}$ *est* sUF-CMVA *sous l'hypothèse que* $\mathsf{MAC}_{\mathsf{BB}}$ *est sUF-CMVA.*

Démonstration. La preuve du Théorème 5 est disponible ci-après à la Section 3.1.2. □

3.1.2 Preuves de sécurité

Nous prouvons ici les Théorèmes 4 et 5 relatifs à la sécurité de nos protocoles. Pour ces preuves comme pour les prochaines, nous reposons sur des réductions (voir Section 1.3.2) à un problème difficile (ici, le problème $q-\mathsf{SDH}-\mathsf{III}$, voir Définition 21) ou aux propriétés des constructions utilisées. Nous présentons les étapes principales de ces preuves.

Preuve de sécurité de $\mathsf{MAC}_{\mathsf{BB}}$

Soit $\mathcal{A}$ un adversaire contre la propriété sUF-CMVA de notre schéma $\mathsf{MAC}_{\mathsf{BB}}$. Il a accès aux deux oracles $\mathcal{O}_{\mathtt{MAC}_{\mathtt{BB}}}$ et $\mathcal{O}_{\mathtt{Verif}_{\mathtt{BB}}}$. Pour tout message m_i choisi, $\mathcal{O}_{\mathtt{MAC}_{\mathtt{BB}}}$ retourne le triplet (A_i, r_i, s_i), pour i dans $\{1,\ldots,q\}$ où q est le nombre maximum de requêtes autorisées pour cet oracle. Pour un quadruplet (A,r,s,m) en entrée, $\mathcal{O}_{\mathtt{Verif}_{\mathtt{BB}}}$ retourne 1 si (A,r,s) est un MAC valide pour m et 0 sinon. $\mathcal{A}$ gagne uniquement si, après au plus q requêtes à l'oracle $\mathcal{O}_{\mathtt{MAC}_{\mathtt{BB}}}$ et q_v requêtes à $\mathcal{O}_{\mathtt{Verif}_{\mathtt{BB}}}$, il parvient à fournir une contrefaçon (A,r,s) pour un message m. Deux types de contrefaçons sont possibles :

Type 1 : l'adversaire $\mathcal{A}$ retourne un triplet (A,r,s) valide sur le message m tel que $(A,r) \neq (A_i, r_i)$, pour tout $i \in \{1,\ldots,q\}$;

Type 2 : l'adversaire $\mathcal{A}$ retourne un triplet (A,r,s) valide sur le message m tel que $(A,r) = (A_j, r_j)$, pour $j \in \{1,\ldots,q\}$, et $(m,s) \neq (m_j, s_j)$.

La réduction $\mathcal{R}$ utilise l'adversaire $\mathcal{A}$ contre l'hypothèse $q-\mathsf{SDH}$ dans les groupes gap-DDH, qui implique l'hypothèse $q-\mathsf{SDH}-\mathsf{III}$ (Section 1.3.1), pour répondre aux challenges fournis par un challenger $\mathcal{C}$. Selon le type d'adversaire, $\mathcal{R}$ fonctionne différemment. Pour commencer, $\mathcal{R}$ choisit donc aléatoirement un bit c_{mode} qui détermine quel type de contrefaçon sera produite.

Type 1. En entrée, $\mathcal{R}$ utilise les paramètres publics (h, g_0, g_1), la valeur publique $Y = g_0^y$ et q triplets aléatoires et distincts (A_i, r_i, s_i) tels que $A_i = (g_1^{m_i} h)^{1/(y+r_i)}$ pour $i \in \{1,\ldots,q\}$, fournis par le challenger $\mathcal{C}$. Le but étant de casser $q-\mathsf{SDH}-\mathsf{III}$, $\mathcal{R}$ a accès à l'oracle $\mathcal{O}_{\mathsf{DDH}}$ qui détermine si (g, h, g^a, h^b) est un quadruplet Diffie-Hellman ou non. $\mathcal{R}$ choisit aléatoirement v dans $\mathbb{Z}_p$, calcule $g = g_1^v$ et fournit l'ensemble des paramètres publics (g, h, g_1, g_0, Y) à $\mathcal{A}$. En utilisant $\mathcal{O}_{\mathsf{DDH}}$, la réduction $\mathcal{R}$ répond aux requêtes de $\mathcal{A}$ comme suit :

- Requêtes $\mathcal{O}_{\mathtt{MAC}_{\mathtt{BB}}}$: étant donné m en entrée, $\mathcal{R}$ choisit l'un des triplets (A_i, r_i, s_i) – qui n'a pas encore été utilisé – et calcule s_i tel que $m + vs_i = m_i$; $\mathcal{R}$ fournit le triplet (A_i, r_i, s_i) qui est alors un MAC valide pour m tel que $A_i^{y+r_i} = g_1^m g^{s_i} h$;
- Requêtes $\mathcal{O}_{\mathtt{Verif}_{\mathtt{BB}}}$: étant donné un quadruplet (A,r,s,m), $\mathcal{R}$ vérifie que $A \neq 1$ puis calcule $B = A^{-r} g_1^m g^s h$; $\mathcal{R}$ fournit (g_0, A, Y, B) en entrée à son oracle $\mathcal{O}_{\mathsf{DDH}}$ et transmet la réponse.

Après au plus q requêtes à $\mathcal{O}_{\mathtt{MAC}_{\mathtt{BB}}}$ et q_v requêtes à $\mathcal{O}_{\mathtt{Verif}_{\mathtt{BB}}}$, l'adversaire $\mathcal{A}$ retourne sa contrefaçon (A,r,s) sur le message m et casse alors la propriété sUF-CMVA de notre schéma $\mathsf{MAC}_{\mathsf{BB}}$. Par conséquent, après avoir calculé $m' = m + sv$, $\mathcal{R}$ obtient une contrefaçon (A,r,m') et est en mesure de casser l'hypothèse $q-\mathsf{SDH}$.

Type 2. Ici, $\mathcal{R}$ utilise l'adversaire $\mathcal{A}$ contre le problème du logarithme discret en réponse au challenge $(g_1, H = g_1^v)$ fourni par $\mathcal{C}$. Son but est de retrouver la valeur v. Pour cela, $\mathcal{R}$ choisit aléatoirement (y, g_0, h) dans $\mathbb{Z}_p \times \mathbb{G}^2$, pose g tel que $g = H$ et calcule $Y = g_0^y$. $\mathcal{R}$ fournit l'ensemble des paramètres publics (g_0, g_1, h, g, Y) à l'adversaire $\mathcal{A}$ et répond aux requêtes de $\mathcal{A}$ comme suit :

- Requêtes $\mathcal{O}_{\mathtt{MAC}_{\mathtt{BB}}}$: étant donné que y lui est connu, $\mathcal{R}$ génère un MAC valide (A, r, s) pour un message m ; $\mathcal{R}$ calcule donc $A = (g_1^m g^s h)^{1/(y+r)}$ où r et s sont aléatoires dans $\mathbb{Z}_p^*$;
- Requêtes $\mathcal{O}_{\mathtt{Verif}_{\mathtt{BB}}}$: étant donné un quadruplet (A, r, s, m), $\mathcal{R}$ calcule $A' = (g_1^m g^s h)^{1/(y+r)}$ et teste si $A = A'$ ou non puis retourne ce résultat à $\mathcal{A}$.

Après au plus q requêtes à $\mathcal{O}_{\mathtt{MAC}_{\mathtt{BB}}}$ et q_v requêtes à $\mathcal{O}_{\mathtt{Verif}_{\mathtt{BB}}}$, l'adversaire $\mathcal{A}$ retourne sa contrefaçon (A, r, s) sur le message m et casse la propriété sUF-CMVA de notre schéma $\mathsf{MAC}_{\mathsf{BB}}$. Par hypothèse, le couple (A, r) est égal à l'une des paires (A_j, r_j) générée par l'oracle $\mathcal{O}_{\mathtt{MAC}_{\mathtt{BB}}}$ suite à une requête de $\mathcal{A}$, pour un $j \in \{1, \ldots, q\}$. Ainsi, $A_j^{y+r_j} = g_1^{m_j} g^{s_j} h = A^{y+r} = g_1^m g^s h$, d'où $g_1^{m_j} g^{s_j} = g_1^m g^s$ et $s_j \neq s$. En effet, dans le cas contraire, m serait égal à m_j ce qui est contradictoire avec l'hypothèse de départ où nous supposons que $(m, s) \neq (m_j, s_j)$. Par conséquent, $g = g_1^{(m-m_j)/(s_j-s)}$ et, en utilisant les valeurs (m, m_j, s, s_j), $\mathcal{R}$ retrouve v qui est donc égal à $(m - m_j)/(s_j - s) \bmod p$ et casse le problème DL. Or, si la réduction est capable de casser DL alors elle est capable de casser $q-\mathsf{SDH}$ en retrouvant le logarithme discret y de $Y = g^y$ en base g.

La réduction $\mathcal{R}$ peut deviner quel type de contrefaçon produit l'adversaire avec une probabilité égale à $1/2$. $\mathcal{R}$ casse donc le problème gap-$q-\mathsf{SDH}$ avec une probabilité $\varepsilon/2$ où ε est la probabilité que $\mathcal{A}$ casse la propriété sUF-CMVA de notre schéma $\mathsf{MAC}_{\mathsf{BB}}$.

Preuve de sécurité – $\mathsf{MAC}^n_{\mathsf{BB}}$

Soit $\mathcal{A}$ un adversaire contre la propriété sUF-CMVA de notre schéma $\mathsf{MAC}^n_{\mathsf{BB}}$. Il a accès aux deux oracles $\mathcal{O}_{\mathtt{MAC}^n_{\mathtt{BB}}}$ et $\mathcal{O}_{\mathtt{Verif}^n_{\mathtt{BB}}}$. Pour tout vecteur de messages $\vec{m}_i = (m_{(i,1)}, \ldots m_{(i,n)})$, $\mathcal{O}_{\mathtt{MAC}^n_{\mathtt{BB}}}$ retourne le triplet (A_i, r_i, s_i), pour i dans $\{1, \ldots, q\}$ où q est le nombre maximum de requêtes autorisées pour cet oracle. Pour un quadruplet, $(A, r, s, \vec{m})$ en entrée, $\mathcal{O}_{\mathtt{Verif}^n_{\mathtt{BB}}}$ retourne 1 si (A, r, s) est un MAC valide pour $\vec{m}$ et 0 sinon. $\mathcal{A}$ gagne uniquement si, après au plus q requêtes à l'oracle $\mathcal{O}_{\mathtt{MAC}^n_{\mathtt{BB}}}$ et q_v requêtes à $\mathcal{O}_{\mathtt{Verif}^n_{\mathtt{BB}}}$, il parvient à fournir une contrefaçon (A, r, s) pour $\vec{m} = (m_1, \ldots, m_n)$. Deux types de contrefaçons sont possibles :

Type 1 : l'adversaire $\mathcal{A}$ retourne un triplet (A, r, s) valide sur $\vec{m}$ tel que $(A, r, s) \neq (A_i, r_i, s_i)$, pour tout $i \in \{1, \ldots, q\}$;

Type 2 : l'adversaire $\mathcal{A}$ retourne un triplet (A, r, s) valide sur $\vec{m}$ tel que $(A, r, s) = (A_j, r_j, s_j)$, pour $j \in \{1, \ldots, q\}$, et $(m_1, \ldots, m_n) \neq (m_{(j,1)}, \ldots, m_{(j,n)})$.

La réduction $\mathcal{R}$ utilise l'adversaire $\mathcal{A}$ contre le schéma $\mathsf{MAC}_{\mathsf{BB}}$, prouvé sUF-CMVA précédemment, pour répondre aux challenges fournis par un challenger $\mathcal{C}$. Selon le type d'adversaire, $\mathcal{R}$ fonctionne différemment. Pour commencer, $\mathcal{R}$ choisit donc aléatoirement un bit c_{mode} qui détermine quel type de contrefaçon est produite.

Type 1. En entrée, $\mathcal{R}$ utilise les paramètres publics (g, h, g_0, g_1) et la valeur publique $Y = g_0^y$ fournis par le challenger $\mathcal{C}$. $\mathcal{R}$ choisit aléatoirement α_i dans $\mathbb{Z}_p^*$, calcule $g_i = g^{\alpha_i}$, pour tout $i \in$

$\{2,\ldots,n\}$ et fournit l'ensemble des paramètres publics $(g,h,g_0,g_1,\{g_i\}_2^n,Y)$ à $\mathcal{A}$. En utilisant ses oracles, la réduction $\mathcal{R}$ répond aux requêtes de $\mathcal{A}$ comme suit :

- Requête $\mathcal{O}_{\mathtt{MAC}^n_{\mathtt{BB}}}$: étant donné $(m_1,\ldots,m_n)$, $\mathcal{R}$ utilise son oracle $\mathcal{O}_{\mathsf{MAC}_{\mathsf{BB}}}$ avec $M = m_1 + \alpha_2 m_2 + \cdots + \alpha_n m_n$ et obtient alors (A,r,s), valide pour $\mathsf{MAC}^n_{\mathsf{BB}}$ qu'il transfert ;
- Requête $\mathcal{O}_{\mathtt{Verif}^n_{\mathtt{BB}}}$: étant donné un quadruplet $(A,r,s,\vec{m})$, $\mathcal{R}$ fournit $M = m_1 + \alpha_2 m_2 + \cdots + \alpha_n m_n$ en entrée à son oracle $\mathcal{O}_{\mathtt{Verif}_{\mathtt{BB}}}$ et transmet la réponse.

Après au plus q requêtes à $\mathcal{O}_{\mathtt{MAC}^n_{\mathtt{BB}}}$ et q_v requêtes à $\mathcal{O}_{\mathtt{Verif}^n_{\mathtt{BB}}}$, l'adversaire $\mathcal{A}$ retourne sa contrefaçon (A,r,s) sur $\vec{m} = (m_1,\ldots,m_n)$ et casse alors la propriété sUF-CMVA de notre schéma $\mathsf{MAC}^n_{\mathsf{BB}}$. Par conséquent, $\mathcal{R}$ obtient une contrefaçon (A,r,s) pour le schéma $\mathsf{MAC}_{\mathsf{BB}}$ sur $M' = m_1 + \alpha_2 m_2 + \cdots + \alpha_n m_n$ et casse la propriété sUF-CMVA de $\mathsf{MAC}_{\mathsf{BB}}$.

Type 2. Ici, la réduction $\mathcal{R}$ utilise l'adversaire $\mathcal{A}$ contre le problème du logarithme discret en réponse au challenge $(g, H = g^v)$ fournit par $\mathcal{C}$. Son but est de retrouver la valeur v. Pour cela, $\mathcal{R}$ choisit aléatoirement (y,g_0,h) dans $\mathbb{Z}_p \times \mathbb{G}^2$ et calcule $Y = g_0^y$. Ensuite, $\mathcal{R}$ choisit un I dans $\{1,\ldots,n\}$ et $(n-1)$ valeurs aléatoires α_i dans $\mathbb{Z}_p^*$. Pour $i \neq I$, $\mathcal{R}$ calcule $g_i = g^{\alpha_i}$ et pose $g_I = H$. $\mathcal{R}$ fournit l'ensemble des paramètres publics à l'adversaire $\mathcal{A}$ et répond à ses requêtes comme suit :

- Requête $\mathcal{O}_{\mathtt{MAC}^n_{\mathtt{BB}}}$: étant donné que y lui est connu, $\mathcal{R}$ génère un MAC valide (A,r,s) pour $(m_1,\ldots,m_n)$; $\mathcal{R}$ calcule donc $A = (g_1^{m_1}\ldots g_n^{m_n} g^s h)^{1/(y+r)}$ avec r et s aléatoires dans $\mathbb{Z}_p^*$;
- Requête $\mathcal{O}_{\mathtt{Verif}^n_{\mathtt{BB}}}$: étant donné un triplet (A,r,s) et $\vec{m} = (m_1,\ldots,m_n)$, $\mathcal{R}$ calcule la valeur $A' = (g_1^{m_1}\ldots g_n^{m_n} g^s h)^{1/(y+r)}$ et teste si $A = A'$ ou non, puis retourne ce résultat.

Après au plus q requêtes à $\mathcal{O}_{\mathtt{MAC}^n_{\mathtt{BB}}}$ et q_v requêtes à $\mathcal{O}_{\mathtt{Verif}^n_{\mathtt{BB}}}$, l'adversaire $\mathcal{A}$ retourne sa contrefaçon (A,r,s) sur $\vec{m} = (m_1,\ldots,m_n)$ et casse alors la propriété sUF-CMVA du schéma $\mathsf{MAC}^n_{\mathsf{BB}}$. Par hypothèse, le triplet (A,r,s) est égal à l'un des triplets (A_j,r_j,s_j) généré par l'oracle $\mathcal{O}_{\mathtt{MAC}^n_{\mathtt{BB}}}$ suite à une requête de $\mathcal{A}$, pour un j dans $\{1,\ldots,q\}$ et tel que $(m_{(j,1)}\ldots m_{(j,n)}) \neq (m_1,\ldots,m_n)$. L'un au moins des éléments est différent dans les deux vecteurs de messages. Avec une probabilité $1/n$, nous avons $m_{(j,I)} \neq m_I$. Ainsi, $g_1^{m_{(j,1)}}\ldots g_n^{m_{(j,n)}} = g_1^{m_1}\ldots g_n^{m_n}$. Par conséquent, le logarithme discret v de $H = g_I$ dans la base g est égal à $\Sigma_{i\neq I}^n \alpha_i \frac{m_i - m_{(j,i)}}{m_{(j,I)} - m_I}$. $\mathcal{R}$ casse le problème DL avec une probabilité ε/n où ε est la probabilité que $\mathcal{A}$ casse la propriété sUF-CMVA de $\mathsf{MAC}^n_{\mathsf{BB}}$. Or, si la réduction est capable de casser DL alors elle est capable la propriété sUF-CMVA de $\mathsf{MAC}_{\mathsf{BB}}$, en retrouvant le logarithme discret y de $Y = g_0^y$ en base g_0.

La réduction $\mathcal{R}$ peut deviner quel type de contrefaçon produit l'adversaire avec une probabilité égale à $1/2$. $\mathcal{R}$ casse donc la propriété sUF-CMVA avec une probabilité $\varepsilon/2n$ dans le pire cas (à savoir la contrefaçon de type 2).

3.2 Notre MAC séquentiellement agrégé

Comme nous l'avons vu à la Section 2.2.1, les signatures séquentiellement agrégées (Définition 31) découlent des signatures agrégées qui sont une variante de la signature numérique classique. Elles permettent à un nombre fini de signataires de signer les uns après les autres leurs messages pour obtenir au final une seule et même signature. En 2008, KATZ et LINDELL [KL08] s'intéressent à une version à clé secrète de cette construction, appelée MAC séquentiellement agrégé.

Nous présentons le modèle de sécurité d'un MAC séquentiellement agrégé, adapté du modèle propre aux signatures [PS16]. Nous détaillons ensuite notre nouveau schéma, basé sur le schéma $\mathsf{MAC}_{\mathsf{GGM}}$ [CMZ14], énoncé à la Section 2.2.2. Nous prouvons également sa sécurité. Cette construction a été publiée au *workshop* VOTING'16, associé à la conférence FC 2016, dans l'article *"Remote Electronic Voting can be Efficient, Verifiable and Coercion-Resistance"* [ABBT16].

3.2.1 Description

Modèle de sécurité

Un schéma de MAC séquentiellement agrégé (`Setup`, `KeyGen`, `AggMAC`,`Verif`) doit satisfaire la propriété de *résistance aux falsifications existentielles contre des attaques adaptatives à messages choisis, avec oracle de vérification*, notée EUF-CMVA. Autrement dit, il doit être impossible de produire un nouveau MAC valide sur un message m qui n'a pas fait l'objet d'une requête de MAC au préalable. Nous présentons ici le modèle dit certifié de LU, OSTROVSKY, SAHAI, SHACHAM et WATERS [LOS+06] où pour chaque requête d'adhésion, l'adversaire doit prouver en plus la connaissance du secret sk associé à la valeur publique $X = h^{sk}$. Cela permet d'extraire la clé sk grâce aux techniques de rejeu et d'extraction de la preuve de connaissance. Plus formellement, la propriété est définie par le jeu suivant entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$:

- **Initialisation :** le challenger $\mathcal{C}$ définie une liste PubList vide pour stocker les futures valeurs publiques, puis exécute `Setup` et `KeyGen` pour obtenir les paramètres publics pp et X^* associé à la clé secrète sk^* ; les valeurs publiques (pp, X^*) sont envoyées à $\mathcal{A}$;
- **Requêtes d'adhésion** $\mathcal{O}_{\texttt{Join}}$ **:** $\mathcal{A}$ envoie des valeurs publiques X_i qui sont ajoutées à PubList ; les clés sk_i correspondantes sont alors extraites ;
- **Requêtes de MAC** $\mathcal{O}_{\texttt{AggMAC}}$ **:** étant donné un MAC agrégé τ_i sur $(m_{(i,1)}, \ldots, m_{(i,r_i)})$ pour $(X_{(i,1)}, \ldots, X_{(i,r_i)})$ dans PubList, $\mathcal{A}$ obtient le MAC agrégé sur $(m_{(i,1)}, \ldots, m_{(i,r_i)}, m_i)$;
- **Sortie :** Après au plus q requêtes à l'oracle de MAC et q_v requêtes à l'oracle de vérification, $\mathcal{A}$ retourne un MAC agrégé τ sur des messages $(m_1^*, \ldots, m_r^*)$. Il remporte le jeu si les conditions suivants sont satisfaites : (1) τ est un MAC agrégé valide pour $(m_1^*, \ldots, m_r^*)$ c'est-à-dire que $\texttt{Verif}(pp, \tau, (m_1^*, \ldots, m_r^*), (sk_1, \ldots, sk_r))$ retourne 1, (2) pour tout $sk_i \neq sk^*$, il existe une valeur publique correspondante X_i dans PubList et (3) il existe un j tel que $sk_j = sk^*$ et $m^*{}_j$ n'a pas été utilisé lors d'une requête à l'oracle de génération de MAC au cours du jeu.

La Figure 3.2 représente la propriété sous forme d'expérience de sécurité, notée $\mathrm{Exp}_{\mathcal{A}}^{\text{EUF-CMVA}}$.

Présentation du schéma

Par soucis de clarté, nous détaillons le fonctionnement de notre schéma dans le cas de deux émetteurs $\mathcal{E}_1$ et $\mathcal{E}_2$. Notre schéma de MAC séquentiellement agrégé est donc défini par les quatre algorithmes suivants.

- $\texttt{Setup}(1^\lambda)$ retourne les paramètres publics du système $pp = (q, \mathbb{G}, g, h, Y_0)$ où $\mathbb{G}$ est un groupe cyclique d'ordre premier q, de longueur λ bits, g, h sont deux générateurs aléatoires de $\mathbb{G}$ tels que $\log_g h$ soit inconnu et $Y_0 = g^{x_0}$ avec x_0 aléatoire dans $\mathbb{Z}_q^*$;
- $\texttt{KeyGen}(pp)$ génère les clés secrètes des émetteurs : le premier émetteur $\mathcal{E}_1$ reçoit $sk_1 = (x_0, x_1)$ et le deuxième émetteur $\mathcal{E}_2$ obtient $sk_2 = x_2$, aléatoires dans $\mathbb{Z}_q^*$; des paramètres

$\mathrm{Exp}_{\mathcal{A}}^{\text{EUF-CMVA}}(1^\lambda)$
1. $pp \leftarrow \mathtt{Setup}(1^\lambda)$, $\varnothing \leftarrow \mathsf{PubList}$;
2. $sk^* \leftarrow \mathtt{KeyGen}(pp)$;
3. $(\tau, (m_1^*, \ldots, m_r^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{AggMAC}}, \mathcal{O}_{\mathtt{Verif}}, \mathcal{O}_{\mathtt{Join}}}(pp, X^*)$;
4. Pour tout $sk_i \neq sk^*$, vérifier que sk_i est associée à un X_i dans $\mathsf{PubList}$;
5. Si, pour tout j tel que $sk_j = sk^*$, m_j^* a été utilisé avec $\mathcal{O}_{\mathtt{MAC}}$, retourner 0 ;
6. Retourner $\mathtt{Verif}(pp, (sk_1, \ldots, sk_r), (m_1^*, \ldots, m_r^*), \tau')$.

FIGURE 3.2 – Résistance aux falsifications

publics associés à ces clés et définis comme $X_1 = h^{x_1}$ et $X_2 = h^{x_2}$ sont ajoutés aux paramètres pp ;

– $\mathtt{AggMAC}(\mathcal{E}_1(pp, sk_1, m_1), \mathcal{E}_2(pp, sk_2, m_2))$ permet de produire un MAC sur les messages m_1 et m_2 créé de façon séquentielle par les deux émetteurs :

Premier émetteur : pour commencer, $\mathcal{E}_1$ génère le MAC $\tau_1 = (u, u')$ où $u = g^{a_1}$ pour a_1 aléatoire dans $\mathbb{Z}_q^*$ et $u' = (g^{m_1 x_1} Y_0)^{a_1} = u^{x_0 + m_1 x_1}$; il envoie ce MAC τ_1 au deuxième émetteur $\mathcal{E}_2$ avec une preuve de connaissance π_1 pour prouver que u' a été correctement calculé :

$$\pi_1 = \mathrm{PoK}\{\alpha, \beta : u = g^\alpha \wedge u' = (g^{m_1 \beta} Y_0)^\alpha \wedge X_1 = h^\beta\}.$$

Deuxième émetteur : Si la preuve π_1 est valide, $\mathcal{E}_2$ utilise τ_1 pour générer le MAC séquentiel $\tau_2 = (w, w') = (u^{a_2}, (u' u^{m_2 x_2})^{a_2})$, avec a_2 aléatoire dans $\mathbb{Z}_q^*$, sur les messages m_1 et m_2, où w' est alors égal à $w^{x_0 + m_1 x_1 + m_2 x_2}$; il construit une preuve de connaissance π_2 pour prouver que τ_2 a été correctement calculé :

$$\pi_2 = \mathrm{PoK}\{\alpha, \beta : w = u^\alpha \wedge w' = (u^{m_2 \beta} u')^\alpha \wedge X_2 = h^\beta\}.$$

– $\mathtt{Verif}(pp, \tau_2, m_1, m_2, sk_1, sk_2)$ teste la validité du MAC τ_2 par rapport aux messages m_1 et m_2 en vérifiant les équations suivantes :

$$w \neq 1 \quad \text{et} \quad w' = w^{x_0 + m_1 x_1 + m_2 x_2}.$$

Généralisation à n émetteurs. Soit $(\mathcal{E}_1, \ldots, \mathcal{E}_n)$ un ensemble de n émetteurs disposant chacun d'une clé secrète $sk_i = x_i$, associée à la valeur publique $X_i = h^{x_i}$, générée par $\mathtt{KeyGen}$. L'émetteur $\mathcal{E}_{i-1}$ envoie à $\mathcal{E}_i$ son MAC séquentiel τ_{i-1} accompagné de la preuve de connaissance π_{i-1} correspondante. À son tour, $\mathcal{E}_i$ génère le MAC séquentiel.

$i$ème émetteur : Si la preuve π_{i-1} est valide, $\mathcal{E}_i$ utilise τ_{i-1} pour générer le MAC séquentiel $\tau_i = (w_i, w_i') = (w_{i-1}^{a_i}, (w_{i-1}' w_{i-1}^{m_i x_i})^{a_i})$ pour a_i aléatoire dans $\mathbb{Z}_q^*$ sur les messages $(m_1, \ldots, m_i)$ tel que w_i' est égal à $w_i^{x_0 + m_1 x_1 + \cdots + m_i x_i}$; il construit également une preuve de connaissance π_i pour attester que τ_i a été correctement calculé :

$$\pi_i = \mathrm{PoK}\{\alpha, \beta : w_i = w_{i-1}^\alpha \wedge w_i' = (w_{i-1}^{m_i \beta} w_{i-1}')^\alpha \wedge X_i = h^{x_i}\}$$

Théorème 6. *Notre schéma de MAC séquentiellement agrégé est* EUF-CMVA*, dans le modèle de l'oracle aléatoire, sous l'hypothèse que* $\mathsf{MAC}_{\mathsf{GGM}}$ *est lui-même UF-CMVA.*

Démonstration. La preuve du Théorème 6 est disponible ci-après à la Section 3.2.2. □

3.2.2 Preuve de sécurité

Nous prouvons ici le Théorème 6 relatif à la sécurité de notre MAC séquentiellement agrégé en construisant une réduction à la sécurité du schéma $\mathsf{MAC}_{\mathsf{GGM}}$, présenté à la Section 2.2.2.

Soit $\mathcal{A}$ un adversaire contre la propriété EUF-CMVA de notre schéma de MAC séquentiellement agrégé. Il a accès à trois oracles $\mathcal{O}_{\texttt{Join}}$, $\mathcal{O}_{\texttt{AggMAC}}$ et $\mathcal{O}_{\texttt{Verif}}$. Pour une valeur publique X_i, $\mathcal{A}$ prouve la connaissance du secret et l'oracle d'adhésion $\mathcal{O}_{\texttt{Join}}$ ajoute alors X_i à PubList. Pour un couple (m,τ) où τ est un MAC agrégé sur $(m_1,\ldots,m_r)$, $\mathcal{O}_{\texttt{AggMAC}}$ retourne le MAC agrégé τ' sur $(m_1,\ldots,m_r,m)$. Enfin, l'oracle $\mathcal{O}_{\texttt{Verif}}$ teste la validité d'un MAC agrégé, retourne 1 s'il est valide et 0 sinon.

La réduction $\mathcal{R}$ utilise l'adversaire $\mathcal{A}$ contre l'hypothèse UF-CMVA du schéma $\mathsf{MAC}_{\mathsf{GGM}}$. En entrée, $\mathcal{R}$ utilise les paramètres publics de $\mathsf{MAC}_{\mathsf{GGM}}$ fournis par le challenger $\mathcal{C}$, à savoir $(pp, C_{x_0} = g^{x_0}h^x, X_1 = h^{x_1})$, et a accès aux deux oracles $\mathcal{O}_{\mathtt{MAC}_{\mathtt{GGM}}}$ et $\mathcal{O}_{\mathtt{Verif}_{\mathtt{GGM}}}$. Ils permettent respectivement d'obtenir et de vérifier un $\mathsf{MAC}_{\mathsf{GGM}}$. Pour commencer, $\mathcal{R}$ fait appel à $\mathcal{O}_{\mathtt{MAC}_{\mathtt{GGM}}}$ sur $m=0$ et reçoit un MAC valide $\tau_0 = (u, u' = u^{x_0})$. $\mathcal{R}$ pose $g = u$, $Y_0 = u'$ et $sk^* = x_1$, inconnu de $\mathcal{R}$, et fournit l'ensemble des valeurs publiques pp et X_1 correspondant à la clé secrète sk^*. Ensuite, $\mathcal{R}$ initialise la liste vide des valeurs publiques PubList et simule les requêtes de $\mathcal{A}$, à partir de ses oracles, comme suit :

- Requêtes d'adhésion $\mathcal{O}_{\texttt{Join}}$: étant donnée une valeur publique X_i, $\mathcal{A}$ doit prouver la connaissance de la clé secrète sk_i correspondante pour ajouter X_i à la liste PubList, ce qui permet à $\mathcal{R}$ d'extraire sk_i grâce à l'extracteur de la preuve de connaissance et d'obtenir toutes les clés secrètes au fil des requêtes ; il ajoute X_i à PubList et stocke la paire (X_i, sk_i) ;
- Requêtes $\mathcal{O}_{\texttt{AggMAC}}$: étant donné $(m_i, \tau_i = (u_i, u_i'))$ où τ_i est un MAC séquentiellement agrégé sur les messages $(m_{(i,1)},\ldots,m_{(i,r)})$ sous les clés secrètes $(sk_{(i,1)},\ldots,sk_{(i,r)})$, $\mathcal{R}$ vérifie la validité de τ_i en calculant $u_i'' = u_i' u_i^{-\sum_{j=1}^{r} sk_{(i,j)} m_{(i,j)}}$ puis appelle l'oracle $\mathcal{O}_{\mathtt{Verif}_{\mathtt{GGM}}}$ sur (u_i, u_i'') pour un message nul ; si la réponse est positive, $\mathcal{R}$ génère un MAC agrégé sur $(m_{(i,1)},\ldots,m_{(i,r)},m_i)$ sous la clé sk_i :
  a) si $sk_i \neq sk^*$ alors $\mathcal{R}$ calcule simplement τ', à partir de la clé secrète sk_i connue, tel que $\tau' = (u_i^t, (u_i' u_i^{m_i sk_i})^t)$ pour t aléatoire dans $\mathbb{Z}_q$ et le transmet à $\mathcal{A}$;
  b) si $sk_i = sk^*$ alors $\mathcal{R}$ utilise l'oracle $\mathcal{O}_{\mathtt{MAC}_{\mathtt{GGM}}}$ sur m_i et obtient $\tau = (u, u') = (u, u^{x_0 + m_i sk^*})$ qu'il transforme en MAC agrégé $\tau' = (u^t, (u' u^{\sum_{j=1}^{r} sk_{i,j} m_{i,j}})^t)$ sur $(m_{(i,1)},\ldots,m_{(i,r)},m_i)$, avec t aléatoire dans $\mathbb{Z}_q$ et le transmet à $\mathcal{A}$;
- Requête $\mathcal{O}_{\texttt{Verif}}$: étant donné $\tau_i = (u_i, u_i')$ sur $(m_{(i,1)},\ldots,m_{(i,r)})$ sous les clés secrètes $(sk_{(i,1)},\ldots,sk_{(i,r)})$ qui correspondent aux valeurs publiques $(X_{(i,1)},\ldots,X_{(i,r)})$, $\mathcal{R}$ teste la validité du MAC :
  a) si, pour tout j de 1 à r, $sk_{(i,j)} \neq sk^*$ alors $\mathcal{R}$ vérifie la validité du MAC à partir de l'algorithme `Verif` pour les messages $(m_{(i,1)},\ldots,m_{(i,r)})$ avec les clés $(sk_{(i,1)},\ldots,sk_{(i,r)})$ qu'il connaît toutes et transmet le résultat à $\mathcal{A}$;

b) si l'une des clés $sk_{(i,j)}$ est sk^* alors $\mathcal{R}$ calcule $u_i'' = u_i' u_i^{-\sum_{j=1, sk_{(i,j)} \neq sk^*}^{r} sk_{(i,j)} m_{(i,j)}}$ puis utilise l'oracle $\mathcal{O}_{\texttt{Verif}_{\texttt{GGM}}}$ sur (u_i, u_i'') pour $\vec{m}_i$ et la clé sk^* et transmet la réponse à $\mathcal{A}$.

Après un nombre fini de requêtes, l'adversaire $\mathcal{A}$ retourne une contrefaçon $\tau^* = (\tau_1^*, \tau_2^*)$ sur $M^* = (m_1^*, \ldots, m_r^*)$, non nuls, sous les clés $SK^* = (sk_1, \ldots, sk_r)$ et casse alors la propriété EUF-CMVA de notre schéma. Ainsi, $\mathcal{R}$ calcule un MAC agrégé valide $\tau' = (\tau_1^*, \tau_2^* \tau_1^{* -\sum_{sk_j \neq sk^*} sk_j m_j^*}) = (\tau_1^*, (\tau_1^*)^{x_0 + m_{j^*}^* x_1})$ sur le message $m_{j^*}^*$ – qui, par définition, n'a jamais été utilisé avec l'oracle $\mathcal{O}_{\texttt{MAC}_{\texttt{GGM}}}$ – et casse alors la propriété UF-CMVA du schéma $\mathsf{MAC}_{\mathsf{GGM}}$.

3.3 Notre signature partiellement aveugle

Les signatures partiellement aveugles (Définition 29) découlent des signatures aveugles qui sont une autre variante de la signature numérique classique. Elles permettent au signataire d'ajouter une information commune à une signature aveugle sur un message m qui lui est inconnu. De cette façon, le signataire garde un minimum de contrôle sur les données qu'il signe. Il peut par exemple ajouter, en accord avec le destinataire, une date, une période de validité ou le montant d'une transaction en cours.

Nous présentons le modèle de sécurité associé à cette primitive cryptographique. Nous détaillons ensuite notre nouveau schéma de signature partiellement aveugle qui est également basée sur la construction introduite par CHASE, MEIKLEJOHN et ZAVERUCHA [CMZ14]. Nous montrons que notre schéma peut être défini sans couplage, sous la forme d'un MAC, puis réalisons les preuves de sécurité associées. Ce résultat a été publié à la conférence FC 2016, dans l'article *"Private eCash in Practice"* [BBD+17].

3.3.1 Description

Modèle de sécurité

Un *schéma de signature partiellement aveugle* est constitué des quatre algorithmes (`Setup`, `KeyGen`, `PartBlindSign`, `Verif`) comme introduits à la Définition29.

OMUF. Une signature partiellement aveugle doit satisfaire la résistance aux falsifications supplémentaires OMUF (*one-more unforgeable*). Autrement dit, même en ayant accès ℓ fois à un oracle qui génère des signatures partiellement aveugles, il doit être impossible de produire $\ell + 1$ signatures valides sur des messages distincts. Plus formellement, la propriété est définie par le jeu suivant entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$:

- **Initialisation :** $\mathcal{C}$ exécute `Setup` et `KeyGen` pour obtenir les paramètres publics pp et la clé publique pk associée à sk ; les valeurs publiques (pp, pk) sont envoyées à $\mathcal{A}$;
- **Requête de signature** $\mathcal{O}_{\texttt{PartBlindSign}}$ **:** $\mathcal{A}$ effectue des requêtes de signatures partiellement aveugles sur une information commune $info$ et $\vec{m} = (m_1, \ldots, m_n)$
- **Sortie :** Après au plus ℓ requêtes à l'oracle de signature partiellement aveugle, $\mathcal{A}$ retourne $\ell + 1$ signatures valides avec une probabilité négligeable. Ces signatures doivent être sur des séquences $\vec{m}_i = (m_{(i,1)}, \ldots, m_{(i,n)})$ de messages distincts et des informations communes $info_i$, pour i allant de 1 à $\ell + 1$.

La Figure 3.3 représente la propriété sous forme d'expérience de sécurité, notée $\mathrm{Exp}_{\mathcal{A}}^{\mathrm{OMUF}}$.

$\mathrm{Exp}_{\mathcal{A}}^{\mathrm{OMUF}}(1^{\lambda})$
1. $pp \leftarrow \mathtt{Setup}(1^{\lambda})$, $\varnothing \leftarrow \mathsf{PubList}$;
2. $sk^{*} \leftarrow \mathtt{KeyGen}(pp)$;
3. $(\sigma_1, \ldots, \sigma_{\ell+1}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{PartBlindSign}}}(pp)$;
4. Si $\mathcal{O}_{\mathtt{PartBlindSign}}$ a été appelé plus de ℓ fois, retourner 0 ;
5. Si il existe un i tel que $\mathtt{Verif}(pp, \vec{m}_i, info_i, \sigma) = \perp$, retourner 0 ;
6. Retourner 1.

FIGURE 3.3 – Résistance aux falsifications supplémentaires

Non-chaînabilité. Une signature partiellement aveugle est dite *non-chaînable* lorsqu'il est impossible de relier entre elles deux signatures ou d'identifier pour qui cette signature a été générée. Cette propriété est propre au caractère "aveugle" de ce type de mécanisme et est parfois notée *blindness*. Nous ne donnerons qu'une preuve rapide de cette propriété.

Présentation du schéma

Notre schéma de signature partiellement aveugle est défini comme suit.

- $\mathtt{Setup}(1^{\lambda})$ retourne les paramètres publics du système $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, \tilde{h})$ c'est-à-dire un environnement bilinéaire de type 3 où q est premier, de longueur λ bits, g, h sont des générateurs aléatoires dans $\mathbb{G}_1$ tels que $\log_g h$ soit inconnu, $\tilde{h}$ est un générateur aléatoire dans $\mathbb{G}_2$ et $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ est une application bilinéaire de type 3 ;
- $\mathtt{KeyGen}(pp)$ génère la clé privée sk du signataire $\mathcal{S}$ égale à $\vec{x} = (x_0, \ldots, x_n)$ où chaque x_i est un élément aléatoire de $\mathbb{Z}_q$ puis construit une mise en gage de Pedersen $C_{x_0} = g^{x_0} h^{\tilde{x}_0}$ sur le secret x_0 avec $\tilde{x}_0$ aléatoire dans $\mathbb{Z}_q$; des paramètres publics $iparams = (C_{x_0}, X_1 = h^{x_1}, \ldots, X_n = h^{x_n}, \tilde{X}_1 = \tilde{h}^{x_1}, \ldots, \tilde{X}_n = \tilde{h}^{x_n})$ sont associés à cette clé privée.
- $\mathtt{PartBlindSign}(\mathcal{R}(pp, info, iparams, \vec{m}), \mathcal{S}(pp, info, sk))$ est un protocole interactif entre le signataire $\mathcal{S}$ et le receveur $\mathcal{R}$ qui souhaite obtenir une signature sur le vecteur de messages $\vec{m} = (m_1, \ldots, m_n)$, sans le dévoiler, et sur une information commune *info*. Le protocole de signature est alors divisé en trois étapes :

  Masquage du message : $\mathcal{R}$ génère une mise en gage de Pedersen $C_{\vec{m}}$ sur $\vec{m}$ et une preuve de connaissance π_1 de ces messages. Il envoie ces deux valeurs au signataire :

  $$C_{\vec{m}} = h^{r} X_1^{m_1} \ldots X_n^{m_n} \quad \text{et} \quad \pi_1 = \mathrm{PoK}\{\alpha_1, \ldots, \alpha_n, \beta : C_{\vec{m}} = h^{\beta} X_1^{\alpha_1} \ldots X_n^{\alpha_n}\}.$$

  Signature de l'engagement : si la preuve π_1 est valide, le signataire $\mathcal{S}$ calcule $u'' = u^{x_0} (C_{\vec{m}} (X_n)^{info})^{b}$ où b est aléatoire dans $\mathbb{Z}_q^{*}$ et $u = h^{b}$; il envoie la signature partiellement aveugle $\sigma'' = (u, u'')$ et la valeur *info* accompagnées d'une preuve de connaissance π_2, prouvant que u'' est bien égal à $u^{x_0 + x_1 m_1 + \cdots + x_{n-1} m_{n-1} + x_n (m_n + info)} h^{br}$

et définie comme suit :

$$\pi_2 = \text{PoK}\{\alpha, \beta, \gamma : u'' = u^{\alpha}(C_{\vec{m}}(X_n)^{info})^{\beta} \wedge C_{x_0} = g^{\alpha}h^{\gamma} \wedge u = h^{\beta}\}$$

Démasquage : pour finir, si π_2 est valide, $\mathcal{R}$ démasque $\sigma'' = (u, u'')$ grâce à l'aléa r utilisé pour l'étape de mise en gage et obtient alors la signature $\sigma = (u, u') = (u, \frac{u''}{u^r}) = (u, u^{x_0+x_1m_1+\cdots+x_{n-1}m_{n-1}+x_n(m_n+info)})$ sur $\vec{m}$ et *info* ;

Randomisation : Pour utiliser cette signature $\sigma = (u, u')$ sans risque, $\mathcal{R}$ la randomise simplement en calculant $\sigma^l = (u^l, (u')^l)$ avec l aléatoire dans $\mathbb{Z}_q^*$;

– `Verif`$(pp, iparams, \vec{m}, info, \sigma')$ teste la validité de la signature $\sigma' = (w, w') = (u^l, (u')^l)$ sur $\vec{m}$ et l'information commune *info*, publiquement vérifiable grâce au couplage e ; il s'assure que $u \neq 1$ et que

$$e(w, \tilde{X}_0) \cdot e(w, \tilde{X}_1)^{m_1} \cdot \cdots \cdot e(w, \tilde{X}_n)^{m_n+info} = e(w', \tilde{h}).$$

Vérification avec clé secrète (MAC). Cette signature partiellement aveugle est construite dans un environnement bilinéaire uniquement pour l'étape publique de vérification. Il est possible de réaliser cette vérification avec la clé secrète. Nous obtenons ainsi un MAC partiellement aveugle. Dans ce cas, les paramètres publics deviennent simplement $pp = (q, \mathbb{G}, g, h)$ et $iparams = (C_{x_0}, X_1, \ldots, X_n)$. La vérification d'une signature $\sigma = (w, w')$ consiste à vérifier que $w \neq 1$ et que $w' = w^{x_0+x_1m_1+\cdots+x_n(m_n+info)}$.

Théorème 7. *Notre schéma de signature partiellement aveugle est* résistant aux falsifications OMUF *et satisfait la propriété de* non-chaînabilité *parfaite, sous l'hypothèse* Assumption1.

Démonstration. La preuve du Théorème 7 est disponible ci-après à la Section 3.3.2. □

3.3.2 Preuves de sécurité

Nous prouvons ici le Théorème 7 relatif à la sécurité de notre signature partiellement aveugle. La première preuve concerne la propriété de résistance aux falsifications supplémentaires pour laquelle nous allons construire une réduction à l'hypothèse Assumption1 (voir Définition 24).

Preuve OMUF

Soit $\mathcal{A}$ un adversaire contre la propriété OMUF. Il a accès à un oracle $\mathcal{O}_{\texttt{PartBlindSign}}$ qui, pour une information commune *info* et une mise en gage de Pedersen $C_{\vec{m}}$ sur les messages $\vec{m} = (m_1, \ldots, m_n)$, retourne une signature partiellement aveugle $\sigma = (u, u'')$ sur $\vec{m}$ et *info*.

La réduction $\mathcal{R}$ utilise l'adversaire $\mathcal{A}$ contre l'hypothèse Assumption1. En entrée, $\mathcal{R}$ utilise les paramètres publics $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, h, \tilde{h})$ et $(X_1, \ldots, X_n, \tilde{X}_1, \ldots, \tilde{X}_n)$ fournis par le challenger $\mathcal{C}$. $\mathcal{R}$ choisit g et C_{x_0} aléatoirement car, pour tout $x_0 \in \mathbb{Z}_q$, il existe un $\tilde{x}_0 \in \mathbb{Z}_q$ tel que $C_{x_0} = g^{x_0}h^{\tilde{x}_0}$. Cette réduction a accès à l'oracle $\mathcal{O}_{\text{Assumption1}}^{(\{X_i\}_i^n, \{\tilde{X}_i\}_i^n)}$ qui prend en entrée la séquence de messages $\vec{m}$ et retourne le couple (u, u') avec $u' = u^{x_0+x_1m_1+\cdots+x_nm_n}$.

$\mathcal{R}$ fournit l'ensemble des paramètres publics à l'adversaire $\mathcal{A}$: $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, \tilde{h})$ et $(C_{x_0}, X_1, \ldots, X_n, \tilde{X}_1, \ldots, \tilde{X}_n)$. En utilisant l'oracle $\mathcal{O}_{\text{Assumption1}}$, $\mathcal{R}$ est en mesure de répondre

aux requêtes de $\mathcal{A}$ à l'oracle $\mathcal{O}_{\texttt{PartBlindSign}}$. Plus précisément, $\mathcal{R}$ utilise les techniques de rejeu et d'extraction de la preuve de connaissance π_1 pour extraire l'aléa r, utilisé pour la mise en gage, et la séquence de messages $\vec{m}$. $\mathcal{R}$ appelle ensuite l'oracle $\mathcal{O}_{\mathsf{Assumption1}}$ sur la séquence $\vec{m}_+ = (m_1, \ldots, m_n, info)$ et obtient une paire (u, u'). Pour obtenir la signature partiellement aveugle, $\mathcal{R}$ utilise l'aléa r et calcule une signature partiellement aveugle (u, u'') sur $\vec{m}$ et *info* telle que $u'' = u'u^r$ et l'envoie, accompagnée de la preuve de connaissance π_2 simulée dans le modèle de l'oracle aléatoire.

Après ℓ appels à l'oracle $\mathcal{O}_{\texttt{PartBlindSign}}$, $\mathcal{A}$ retourne $\ell+1$ signatures valides sur $(\sigma_i, info_i, \vec{m}_i)$ tels que $\vec{m}_i + \overrightarrow{info}_i \neq \vec{m}_j + \overrightarrow{info}_j$ pour tout $j \neq i$ dans $\{1, \ldots, \ell+1\}$ avec $\overrightarrow{info}_k = (\underbrace{0, 0, \ldots, 0}_{\text{n-1 fois}}, info_k)$. Il casse alors la propriété OMUF du schéma. Par conséquent, $\mathcal{R}$ détient $\ell+1$ signatures sur des messages distincts et est en mesure de casser l'hypothèse Assumption1.

Preuve de non-chaînabilité

Au cours d'une exécution du protocole `PartBlindSign`, les données dont dispose le signataire $\mathcal{S}$ sont (1) une mise en gage $C_{\vec{m}} = h^r X_1^{m_1} \ldots X_n^{m_n}$, (2) une preuve π_1 de cette mise en gage et (3) la signature partiellement aveugle (u, u'') qu'il a calculée.

Pour prouver la non-chaînabilité de ce protocole, il faut étudier ce que pourrait utiliser un adversaire $\mathcal{A}$ contre cette propriété à partir de ces trois éléments. (1) Par définition, la mise en gage de Pedersen assure l'indistinguabilité parfaite : elle ne révèle donc aucune information sur le message. (2) De même, π_1 qui est une preuve de connaissance à divulgation nulle de connaissance n'offre, par définition, aucune information sur les secrets sous-jacents. (3) Seule la signature pourrait aider l'adversaire $\mathcal{A}$ à relier une paire (messages, signature) à une exécution donnée du protocole.

Soit (v, v') une signature valide sur un vecteur de messages $\vec{m'} = (m'_1, \ldots, m'_n)$ et *info'* distincts des messages de $\vec{m}$ et de *info*. Nous considérons une version randomisée $(w = v^l, w' = (v')^l)$ de cette signature. Alors, il existe un l_0 tel que $w = u^{l_0}$ et un r_0 tel que $C_{\vec{m}} = h^{r_0} X_1^{m_1} \ldots X_n^{m_n}$. Pour $u' = \frac{u''}{u^{r_0}}$ alors cela implique que $w' = (u')^{l_0}$. De cette façon, (w, w') aurait très bien pu être générée durant cette même exécution qui a permis d'obtenir la mise en gage $C_{\vec{m}}$, la preuve π_1 et la signature (u, u''). Par conséquent, disposer d'une signature partiellement aveugle ne permet pas de la relier à une exécution particulière. Notre schéma respecte alors la propriété d'indistinguabilité parfaite.

Conclusion

Dans ce chapitre, nous avons présenté trois nouvelles primitives cryptographiques. Pour chacune d'elles, nous avons défini le modèle de sécurité et prouvé leur sécurité. Ces trois nouvelles primitives revêtent un intérêt propre mais peuvent être utilisées pour construire de nouveaux systèmes respectueux de la vie privée.

Tout d'abord, nous présentons un schéma d'accréditations anonymes avec vérification par clé, basé sur notre construction nommée $\mathsf{MAC}^n_{\mathsf{BB}}$. Ensuite, nous proposons deux systèmes qui peuvent être assimilés à ce type d'accréditation. Nous traitons ainsi le cas du vote électronique grâce à notre MAC séquentiellement agrégé puis le cas du paiement électronique dit privé en utilisant notre signature partiellement aveugle.

Chapitre 4

Conception d'un système d'accréditations anonymes avec vérification par clé

Dans ce chapitre, nous présentons les accréditations anonymes avec vérifiaction par clé. Nous introduisons alors un nouveau schéma basé sur notre MAC algébrique $\mathsf{MAC}^n_{\mathsf{BB}}$ détaillé à la Section 3.1.1. Ce système permet d'atteindre la propriété de non-chaînabilité pour une accréditation utilisée plusieurs fois tout en étant efficace et adapté aux périphériques ayant des capacités limitées. En outre, une légère modification côté vérificateur permet d'adapter cette contribution en un système vérifiable publiquement grâce à l'utilisation de couplages.

Ce résultat a été publié à la conférence SAC 2016, dans l'article intitulé *"Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials"* [BBDT16] en même temps que les MACs algébriques $\mathsf{MAC}_{\mathsf{BB}}$ et $\mathsf{MAC}^n_{\mathsf{BB}}$ présentés précédemment.

Sommaire

4.1 Accréditations anonymes avec vérification par clé

Les accréditations sont des attestions personnelles permettant à un utilisateur de convaincre autrui qu'il possède une autorisation ou une qualification particulières. Les diplômes, les cartes étudiants ou encore les permis de conduire sont autant d'exemples d'accréditations classiques. Elles sont propres à un individu et générées par une entité de confiance. Cependant, ces accréditations contiennent des informations à caractère personnel comme un nom, une adresse, une date de naissance ou encore des données biométriques.

Dans cette section, nous présentons l'état de l'art des accréditations anonymes et plus particulièrement leur variante "avec vérification par clé". Nous en présentons la définition et le modèle de sécurité.

4.1.1 État de l'art

En 1982, CHAUM [Cha82] introduit l'idée des systèmes d'accréditations anonymes. Un tel système permet à un utilisateur de prouver que ses attributs ont été certifiés, sans révéler d'information superflue. Idéalement, l'utilisation successive d'une même accréditation ne doit pas pouvoir être détectée, pour éviter tout risque de traçage. En outre, il peut parfois s'avérer utile de certifier certains attributs sans les révéler. Un tel système requiert un schéma de signature qui permet d'une part de signer des messages mis en gage et d'autre part de prouver efficacement la connaissance d'une signature sans la révéler. De nombreuses applications existent pour les accréditations anonymes comme le paiement [Cha82] et le vote électronique. Chacune d'entre elles entraîne ses propres contraintes et nécessite généralement des solutions efficaces qui doivent être adaptées aux environnements où elles peuvent être déployées. Nous reviendrons sur ces cas d'usages et leurs spécificités plus en détails dans les chapitres suivants.

En 2013, au sein de Microsoft, PAQUIN et ZAVERUCHA [Paq13, PZ13] proposent l'un des systèmes d'accréditations anonymes les plus connus actuellement, appelé *U-Prove* et basé sur un schéma de signature aveugle due à BRANDS [Bra94]. Cette construction est particulièrement efficace puisqu'elle repose sur des groupes d'ordre premier et permet de choisir les attributs communiqués. Néanmoins, *U-Prove* ne permet pas la non-chaînabilité lorsqu'une même accréditation est utilisée à plusieurs reprises. Pour atteindre cette propriété, l'utilisateur doit détenir une accréditation différente pour chaque nouvelle utilisation. En outre, la sécurité de ce schéma n'a pas été prouvée formellement, à ce jour.

La même année, BALDIMTSI et LYSYANSKAYA [BL13] proposent un système un peu moins efficace qui repose sur une extension d'une signature proposée par ABE [Abe01]. Ce schéma est prouvé sûr dans le modèle de l'oracle aléatoire sous l'hypothèse DDH. Plus récemment, en 2015, FUCHSBAUER, HANSER et SLAMANIG [FHS15] introduisent un schéma prouvé sûr dans le modèle standard. Cependant, tout comme *U-Prove*, une accréditation ne peut servir qu'une seule fois au risque de devenir traçable.

Le second système d'accréditations anonymes parmi les plus répandus est celui proposé par IBM en 2010 [IBM10], appelé *Identity Mixer* ou plus couramment *Idemix*. Il est basé sur un schéma de signature proposé par CAMENISCH et LYSYANSKAYA (CL) [CL01, CL03]. Ce système satisfait la propriété de non-chaînabilité multiple mais cela entraîne des pertes de performances car les signatures CL reposent sur l'hypothèse RSA forte [BP97]. Ainsi, les paramètres RSA requis sont particulièrement grands et rendent *Idemix* inadapté pour les périphériques aux ressources limitées. Néanmoins, en 2013, VULLERS et ALPÁR [VA13] réalisent une implémentation d'*Idemix* sur une carte à puces MULTOS. Pour un module de 1024 bits, leur implémentation nécessite une seconde pour présenter une accréditation avec trois attributs dont un seul reste caché. En 2014, DE LA PIEDRA, HOEPMAN et VULLERS [PHV14] s'intéressent aux capacités limitées de la mémoire vive (RAM, pour *Random Access Memory*) des cartes à puces et proposent une implémentation plus efficace pour *Idemix*. Grâce à ce travail, une carte à puce peut supporter des accréditations pour plus de cinq attributs. Malgré cette amélioration, les

temps d'exécution restent trop importants pour certains cas d'usages, ce qui limite l'utilisation d'*Idemix* en pratique.

CAMENISCH et LYSYANSKAYA [CL04] introduisent en 2004 un schéma de signature efficace dans un environnement bilinéaire et l'utilise pour créer un schéma d'accréditations anonymes. Par la suite, AKAGI, MANABE et OKAMOTO [AMO08] proposent un schéma plus efficace basé sur les signatures Boneh-Boyen (voir Section 2.2.1). En 2015, CAMENISCH, DUBOVITSKAYA, HARALAMBIEV et KOHLWEISS [CDHK15] définissent un nouveau schéma d'accréditations anonymes prouvé sûr dans le modèle UC (*Universal Composability*, [Can01]). Il satisfait la non-chaînabilité multiple et assure une taille constante pour la preuve de présentation de l'accréditation. Cependant, toutes ces constructions impliquent des calculs de couplages de la part de l'utilisateur ou bien des calculs dans le groupe $\mathbb{G}_2$. Par conséquent, ces systèmes ne peuvent pas être implémentés sur les cartes SIM actuelles puisqu'elles ne supportent pas ce type d'opérations.

En 2014, CHASE, MEIKLEJOHN et ZAVERUCHA [CMZ14] introduisent les accréditations anonymes dites "avec vérification par clé". Ce nouveau type de schéma ne permet plus la vérification publique d'une accréditation mais offre en contrepartie de meilleures performances. Cela répond à des problématiques bien précises où l'émetteur et le vérificateur possèdent une même clé secrète commune et repose sur des MACs algébriques. En outre, les auteurs mettent en avant le fait qu'il est possible de passer d'un système à clé publique à une version à clé secrète. Pour cela, il suffit d'utiliser son accréditation anonyme classique, publique, auprès d'un vérificateur qui va alors la convertir en une accréditation vérifiable uniquement par lui pour faciliter leurs futurs échanges. Ils proposent alors deux nouveaux schémas de MACs algébriques, notés $\mathsf{MAC}_{\mathsf{DDH}}$ et $\mathsf{MAC}_{\mathsf{GGM}}$ (voir Section 2.2.2), à partir desquels ils proposent des schémas d'accréditations anonymes avec vérification par clé. Cependant, pour n attributs non-révélés, chaque construction a une complexité en $O(n)$ en nombre d'éléments du groupe cyclique sous-jacent car chaque nouvel attribut non-révélé nécessite une clé secrète supplémentaire. En outre, la garantie d'anonymat procurée n'est pas parfaite mais calculatoire puisque chacun des attributs est caché séparément avec du chiffrement ElGamal.

4.1.2 Définition

Un système d'accréditation anonyme avec vérification par clé fait interagir un utilisateur $\mathcal{U}$, un émetteur $\mathcal{E}$ et un vérificateur $\mathcal{V}$. La particularité est que l'émetteur et le vérificateur disposent d'une clé secrète commune. De façon informelle, les propriétés attendues sont les suivantes :

- une accréditation anonyme valide sera toujours acceptée par le vérificateur ;
- il doit être impossible de convaincre de la possession d'une accréditation valide si ce n'est pas le cas ;
- l'accréditation ne doit révéler aucune information supplémentaire sur les attributs certifiés.

Définition 38. Un *schéma d'accréditations anonymes avec vérification par clé*, noté KVAC pour *Keyed-Verification Anonymous Credentials*, est constitué des quatre étapes associées aux algorithmes suivants :

Initialisation. L'initialisation du système est réalisée par l'algorithme `Setup`. Il prend en entrée 1^λ où λ est un paramètre de sécurité et retourne les paramètres publics du système, notés pp.

$$pp \leftarrow \texttt{Setup}(1^\lambda)$$

Génération des clés. L'algorithme de génération `CredKeyGen` des clés est probabiliste. Il prend en entrée les paramètres publics pp et génère une clé secrète sk pour l'émetteur – qu'il partage avec le vérificateur $\mathcal{V}$ – et la clé publique pk correspondante.

$$(pk, sk) \leftarrow \texttt{CredKeyGen}(pp)$$

Accréditation anonyme. L'accréditation anonyme est obtenue lors d'un protocole interactif, noté `BlindIssue`, entre un utilisateur $\mathcal{U}$ et l'émetteur $\mathcal{E}$. $\mathcal{U}$ connaît la clé publique pk et souhaite obtenir une accréditation anonyme sur un ensemble d'attributs $\vec{m} = (m_1, \ldots, m_n)$ et une valeur secrète s, sans les révéler. L'émetteur $\mathcal{E}$ utilise sa clé secrète sk pour fournir une accréditation σ à l'utilisateur $\mathcal{U}$.

$$\sigma \leftarrow \texttt{BlindIssue}(\mathcal{U}(pp, pk, \vec{m}, s), \mathcal{E}(pp, sk))$$

Présentation de l'accréditation. Pour prouver qu'il détient une accréditation anonyme sur ses attributs $\vec{m}$ qui vérifient un prédicat ϕ, l'utilisateur $\mathcal{U}$ engage un protocole interactif avec un vérificateur $\mathcal{V}$ qui utilise sa clé secrète sk pour la vérification. Si l'accréditation est valide, il retourne 1 et 0 sinon.

$$\{0, 1\} \leftarrow \texttt{Show}(\mathcal{U}(pp, pk, \vec{m}, s, \sigma, \phi), \mathcal{E}(pp, sk, \phi))$$

Modèle de sécurité

Un schéma d'accréditation anonyme avec vérification par clé, défini par (`CredKeyGen`, `CredVerif`, `Issue`, `BlindIssue`, `Show`, `ShowVerif`), est dit sûr s'il vérifie les propriétés de *complétude*, de *résistance aux falsifications*, d'*anonymat*, d'*émission aveugle* et de *consistance des paramètres* définies ci-après.

Nous reprenons ici le modèle introduit en même tant que les KVAC dans [CMZ14]. En accord avec cette modélisation et par souci de clarté, nous présentons les propriétés dans le cas spécifique où les attributs associés à l'accréditation fournie sont connus de l'émetteur. Pour cela, il est nécessaire d'avoir un protocole bipartite supplémentaire qui permet à un utilisateur d'obtenir une accréditation sur des messages révélés, contrairement au cas de `BlindIssue`. Nous ajoutons donc les algorithmes suivants à la Définition 38 d'un KVAC :

Accréditation. L'accréditation est générée par l'algorithme `Issue` sur un vecteur d'attributs $\vec{m} = (m_1, \ldots, m_n)$, en utilisant la clé secrète sk. Dans le cas où l'émetteur $\mathcal{E}$ n'est pas de confiance, il faut utiliser `BlindIssue` sur des attributs cachés.

$$\sigma \leftarrow \texttt{Issue}(pp, \vec{m}, sk)$$

Vérification. L'accréditation σ est vérifiée, en utilisant la clé secrète sk, par rapport aux attributs $\vec{m}$. Cette étape nécessite de connaître à la fois l'accréditation et les attributs as-

sociés, l'algorithme CredVerif n'est donc jamais appelé en pratique. En revanche, il est utilisé pour définir l'ensemble des accréditations valides pour les attributs $(m_1, \ldots, m_n)$ sous la clé sk.

$$\{0,1\} \leftarrow \texttt{CredVerif}(pp, sk, \vec{m}, \sigma)$$

Par la suite, la notation ShowVerif réfère à la partie exécutée par l'émetteur dans l'étape de présentation de l'accréditation Show. Nous notons Φ l'ensemble de prédicats acceptés par le système d'accréditation et *Att* l'espace des attributs.

Complétude. La propriété de *complétude* (*correctness*) assure qu'une accréditation correctement générée est acceptée par le vérificateur. Plus formellement, un schéma de KVAC satisfait la *complétude*, si pour tout $(m_1, \ldots, m_n)$ dans *Att*, pour tout λ suffisamment grand,

$$\mathbb{P}\,(\, pp \xleftarrow{R} \texttt{Setup}(1^\lambda); (sk, pk) \xleftarrow{R} \texttt{CredKeyGen}(pp);\\ \sigma \leftarrow \texttt{Issue}(sk, (m_1, \ldots, m_n)) : \texttt{CredVerif}(sk, (m_1, \ldots, m_n), \sigma) = 0\,) = 0$$

et, pour tout $\phi \in \Phi$, $(m_1, \ldots, m_n) \in Att$ tel que $\phi(m_1, \ldots, m_n) = 1$,

$$\mathbb{P}\,(\, pp \xleftarrow{R} \texttt{Setup}(1^\lambda); (sk, pk) \xleftarrow{R} \texttt{CredKeyGen}(pp); \sigma \xleftarrow{R} \texttt{Issue}(sk, (m_1, \ldots, m_n))\\ \texttt{Show}(pk, \sigma, (m_1, \ldots, m_n), \phi) \leftrightarrow \texttt{ShowVerif}(sk, \phi) \rightarrow b : b = 0\,) = 0$$

Non-falsification. La propriété de résistance aux falsifications (*Unforgeable*) assure qu'il est difficile de créer une contrefaçon valide. Plus formellement, pour un schéma de KVAC, le protocole de présentation de l'accréditation Show défini par (CredKeyGen, Issue) est *résistant aux falsifications*, si pour tout adversaire $\mathcal{A}$, il existe une fonction négligeable ν telle que, pour tout paramètre de sécurité λ :

$$\mathbb{P}\,(\, pp \xleftarrow{R} \texttt{Setup}(1^\lambda); (pk, sk) \xleftarrow{R} \texttt{CredKeyGen}(pp);\\ (state, \phi) \leftarrow \mathcal{A}(pp, pk)^{\mathcal{O}_{\texttt{Issue}}(sk,\cdot), \mathcal{O}_{\texttt{ShowVerif}}(sk,\cdot)};\\ \mathcal{A}(state) \leftrightarrow \texttt{ShowVerif}(sk, \phi) \rightarrow b \text{ tel que}\\ b = 1 \wedge (\forall (m_1, \ldots, m_n) \in Q, \phi(m_1, \ldots, m_n) = 0)\,) = \nu(\lambda)$$

où Q est la liste de l'ensemble des attributs $(m_1, \ldots, m_n)$ utilisés pour les requêtes à l'oracle $\mathcal{O}_{\texttt{Issue}}(sk, \cdot)$ et où toutes les exécutions de l'oracle $\mathcal{O}_{\texttt{ShowVerif}}$ sont séquentielles.

Anonymat. La propriété d'anonymat garantit qu'aucune information personnelle n'est divulguée au cours du système. Plus formellement, pour un schéma de KVAC, le protocole de présentation de l'accréditation Show défini par (CredKeyGen, Issue) est *anonyme*, si pour tout adversaire $\mathcal{A}$, il existe un algorithme efficace SimShow et une fonction négligeable ν tel que, pour tout λ, pour tout $\phi \in \Phi$ et $(m_1, \ldots, m_n)$ dans *Att* tel que $\phi(m_1, \ldots, m_n) = 1$, pour tout paramètres publics pp, toutes paires de clés (pk, sk) et tout accréditation σ valide :

$$\{\, \texttt{Show}(pk, \sigma, (m_1, \ldots, m_n), \phi) \leftrightarrow \mathcal{A} \rightarrow state \,\} \approx \{\, \texttt{SimShow}(pk, sk, \phi) \,\}$$

Autrement dit, la vue de l'adversaire $\mathcal{A}$ pendant l'exécution de Show peut être simulée

par SimShow qui a seulement accès à ϕ et une clé secrète sk valide qui correspond à la clé publique pk.

Émission aveugle (*Blind Issuance*). Soit $\mathcal{U}$ un utilisateur qui souhaite obtenir une accréditation sur des attributs $(m_1, \ldots, m_n)$ par un émetteur $\mathcal{E}$ qui ne connaît qu'un sous-ensemble S de ces attributs. Nous considérons une fonction f définie par $f((S, pp, pk), (sk, r), (m_1, \ldots, m_n))$, sur l'entrée partagée (S, pp, pk), l'entrée (sk, r) de l'émetteur $\mathcal{E}$ et l'entrée $(m_1, \ldots, m_n)$ de l'utilisateur ; f retourne :

- $\perp$ à l'émetteur $\mathcal{E}$ et "erreur sur pp" à l'utilisateur $\mathcal{U}$ si (pk, sk) n'appartient pas aux données CredKeyGen(pp) ;
- $\perp$ à l'émetteur $\mathcal{E}$ et "erreur sur les attributs" à l'utilisateur $\mathcal{U}$ si S n'est pas conforme avec $(m_1, \ldots, m_n)$;
- $\sigma \leftarrow$ Issue$(pp, (m_1, \ldots, m_n), sk; r)$ où Issue$(pp, (m_1, \ldots, m_n), sk; r)$ correspond à l'exécution de Issue$(pp, (m_1, \ldots, m_n), sk)$ avec l'aléa r.

Le protocole BlindIssue satisfait la propriété d'*émission aveugle* pour Issue si c'est un protocole de calcul bipartite sûr [Gol04], contre des adversaires dit malveillants, pour la fonction précédente.

Consistance des paramètres. L'algorithme de génération des paramètres CredKeyGen satisfait la propriété de *consistance des paramètres* si, pour tout adversaire $\mathcal{A}$ connaissant les paramètres publics pp générés par l'exécution de Setup(1^λ), la probabilité que $\mathcal{A}$ produise (pk, sk_1, sk_2) tel que (pk, sk_1) et (pk, sk_2) sont deux couples possibles de CredKeyGen(pp) est négligeable.

Les propriétés de *complétude*, d'*émission aveugle* et de *consistance des paramètres* assurent que si un utilisateur reçoit une accréditation produite par BlindIssue alors cette accréditation est acceptée par CredVerif pour la clé secrète sk connue de l'émetteur et associée à la clé publique pk. En outre, la propriété d'*anonymat* garantit que Show ne permet pas à l'émetteur d'apprendre d'autres informations sur ces attributs hormis le fait qu'ils satisfont le prédicat ϕ.

4.2 Notre système

Dans cette section, nous présentons notre nouveau schéma qui offre une efficacité comparable aux meilleurs schémas actuels mais en assurant une propriété supplémentaire. Bien que reposant sur une primitive à clé secrète, nous montrons également qu'il peut être modifié simplement pour permettre les vérifications publiques.

4.2.1 Description

Notre schéma d'accréditations anonymes avec vérification par clé repose sur notre construction $\mathsf{MAC}^n_{\mathsf{BB}}$, présentée à la Section 3.1.1, et est défini par les quatre étapes suivantes. Il fait intervenir un utilisateur $\mathcal{U}$ qui souhaite obtenir une accréditation anonyme sur un ensemble d'attributs, un émetteur $\mathcal{E}$ qui génère l'accréditation attendue et un vérificateur $\mathcal{V}$ qui s'assure de sa validité lorsqu'elle est utilisée.

Initialisation

L'algorithme `Setup` génère les paramètres publics $pp = (p, \mathbb{G}, g_0, g_1, g_2, \ldots, g_n, g, h, f)$ où $\mathbb{G}$ est un groupe cyclique d'ordre premier p, de longueur λ bits, et $(h, g, g_0, \{g_i\}_{i=1}^n, f)$ sont des générateurs aléatoires de $\mathbb{G}$ pour lequel le problème DDH est difficile (voir Définition 17).

Pour i allant de 1 à n, chaque g_i est associé à un type particulier d'attributs comme l'âge du destinataire ou son genre par exemple. Ceci permet de distinguer clairement chaque attribut et d'éviter toute ambiguïté.

Par la suite, l'ensemble des calculs sur les exposants seront réalisés **modulo p** même si ce n'est pas toujours précisé explicitement.

Génération des clés

L'algorithme `CredKeyGen` génère les clés des différents participants. Pour l'émetteur $\mathcal{E}$, la clé secrète est y aléatoire dans $\mathbb{Z}_p^*$ et est associée à la valeur publique $Y = g_0^y$. La clé secrète y est également communiquée au vérificateur $\mathcal{V}$. Enfin, chaque utilisateur $\mathcal{U}$ du système reçoit une clé secrète notée sk_u et une clé publique correspondante notée pk_u. Cette clé publique est utilisée pour l'authentifier lorsqu'il souhaite obtenir une accréditation anonyme.

Accréditation anonyme

Pour obtenir une accréditation sur la séquence d'attributs $\vec{m} = (m_1, \ldots, m_n)$, l'émetteur $\mathcal{E}$ et l'utilisateur $\mathcal{U}$ s'engagent dans le protocole interactif `BlindIssue`. La Figure 4.1 présente les différentes étapes de ce protocole, détaillées ci-après.

Après s'être authentifié avec sa clé publique pk_u, l'utilisateur $\mathcal{U}$ construit la mise en gage de Pedersen $C_{\vec{m}} = g_1^{m_1} \ldots g_n^{m_n} g^s$ (voir Définition 33), sur la séquence d'attributs $\vec{m}$ qu'il souhaite garder privée, avec s aléatoire dans $\mathbb{Z}_p^*$. Pour prouver que cette mise en gage a été construite correctement, il génère une preuve de connaissance π_1 (voir Définition 35). L'utilisateur $\mathcal{U}$ envoie à l'émetteur $\mathcal{E}$ sa mise en gage $C_{\vec{m}}$ accompagnée de π_1 définie comme suit :

$$\pi_1 = \text{PoK}\{\alpha_1, \ldots, \alpha_{n+1} : C_{\vec{m}} = g_1^{\alpha_1} g_2^{\alpha_2} \ldots g_n^{\alpha_n} g^{\alpha_{n+1}}\}.$$

Ensuite, $\mathcal{E}$ teste la validité de la preuve π_1. Si elle est correcte, il calcule la valeur suivante pour r et s' tirés aléatoirement dans $\mathbb{Z}_p$:

$$A = (C_{\vec{m}} \cdot g^{s'} \cdot h)^{1/(y+r)}.$$

Il obtient alors un $\mathsf{MAC}_{\mathsf{BB}}^n$ sur $(m_1, \ldots, m_n)$. Pour prouver que cette valeur a été correctement calculée, il construit une preuve π_2. Il transmet le triplet (A, r, s') à l'utilisateur $\mathcal{U}$, accompagné de π_2 définie comme suit :

$$\pi_2 = \text{PoK}\{\gamma : B = A^\gamma \wedge Y = g_0{}^\gamma\} \text{ avec } B = C_{\vec{m}} \cdot g^{s'} \cdot h \cdot A^{-r} = A^y.$$

Après réception, $\mathcal{U}$ teste la validité de la preuve π_2. Si elle est valide, il calcule $\tilde{C}_{\vec{m}} = C_{\vec{m}} g^{s'} h$ et le secret $s_u = s + s'$ qui n'est connu que de lui seul. Enfin, il obtient son accréditation anonyme σ, égale à (A, r, s_u), sur $\vec{m}$.

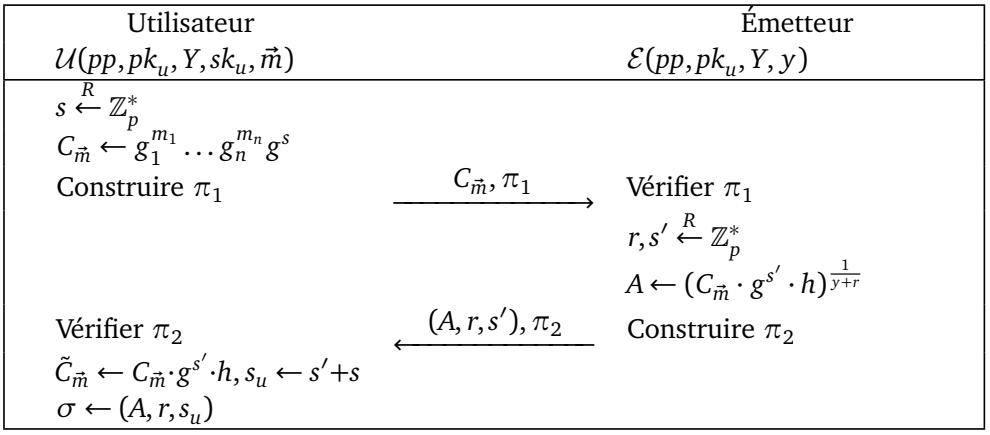


FIGURE 4.1 – Algorithme `BlindIssue` de notre schéma d'accréditations anonymes

Dans le cas où l'utilisateur $\mathcal{U}$ accepte de révéler l'ensemble de ses attributs ou seulement un sous-ensemble d'entre eux, il les envoie simplement sans passer par l'étape de mise en gage ou l'applique uniquement à ceux qu'il souhaite cacher.

Présentation de l'accréditation

Pour attester de façon anonyme qu'il dispose d'une accréditation, l'utilisateur $\mathcal{U}$ s'engage dans le protocole interactif `Show` avec le vérificateur $\mathcal{V}$. La Figure 4.2 reprend les différentes étapes de ce protocole, détaillées ci-après.

Pour commencer, $\mathcal{U}$ calcule une version randomisée B_0 de son accréditation avec un l aléatoire dans $\mathbb{Z}_p^*$, une valeur C qui intègre cette accréditation et ses attributs ainsi qu'une valeur E pour prouver que l'aléa l choisi est non nul. Pour t aléatoire dans $\mathbb{Z}_p^*$, les valeurs calculées sont :

$$B_0 = A^l, \quad C = \tilde{C}_{\vec{m}}^{\,l} \cdot B_0^{-r} \quad \text{et} \quad E = C^{\frac{1}{l}} f^t.$$

En partant de l'accréditation initiale A, nous avons alors les relations suivantes :

$$\begin{aligned}
& A^{y+r} &=& \; C_{\vec{m}} g^{s'} h \\
\Rightarrow\; & A^{y+r} &=& \; g_1^{m_1} g_2^{m_2} \dots g_n^{m_n} g^{s_u} h \\
\Rightarrow\; & (A^{y+r})^l &=& \; g_1^{lm_1} \dots g_n^{lm_n} g^{ls_u} h^l \\
\Rightarrow\; & A^{ly} &=& \; g_1^{lm_1} \dots g_n^{lm_n} g^{ls_u} h^l \cdot A^{-lr} \\
\Rightarrow\; & B_0^y &=& \; \tilde{C}_{\vec{m}}^{\,l} B_0^{-r} \quad = \quad C
\end{aligned}$$

L'utilisateur $\mathcal{U}$ envoie (B_0, C, E) au vérificateur $\mathcal{V}$, accompagné d'une preuve de connaissance π_3 qui assure qu'il possède une accréditation valide pour laquelle il connaît les secrets associés et que la valeur mise en gage dans E est bien différente de zéro [Bra97]. La preuve π_3 est définie comme suit :

$$\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \dots, \delta_{n+1}, \gamma, \theta : E = C^\alpha f^\beta \wedge E \cdot h^{-1} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot B_0^\lambda \cdot f^\beta \wedge C = E^\theta f^\gamma\}$$

Après réception, le vérificateur $\mathcal{V}$ calcule $C' = B_0^y$ et vérifie que la valeur C' est bien égale

à C. Il vérifie ensuite la preuve π_3. Si ces vérifications sont valides, il est alors assuré que $\mathcal{U}$ détient une accréditation valide sur des attributs $(m_1, \ldots, m_n)$.

Utilisateur $\mathcal{U}(pp, (A, r, s_u), \tilde{C}_{\vec{m}}, \vec{m})$		Vérificateur $\mathcal{V}(pp, y)$
$l, t \xleftarrow{R} \mathbb{Z}_p^*$ $B_0 \leftarrow A^l$ $C \leftarrow \tilde{C}_{\vec{m}}^l B_0^{-r}$ $E \leftarrow C^{1\backslash l} f^t$ Construire π_3	$\xrightarrow{(B_0, C, E), \pi_3}$	$C' \leftarrow B_0^y$ Tester si $C = C'$ Vérifier π_3

FIGURE 4.2 – Algorithme Show de notre schéma d'accréditations anonymes

Nous présentons plus en détails la preuve π_3 ci-après.

Détails de la preuve π_3. La preuve de connaissance π_3 est calculée par l'utilisateur qui joue alors le rôle du prouveur. Le but est de prouver qu'il possède une accréditation valide, que la valeur mise en gage dans E est bien différente de 0 et qu'il connaît les secrets associés. La Figure 4.3 détaille les différents calculs réalisés au cours de cette preuve.

Prouveur $\mathcal{U}(pk, (B_0, C, E), (m_1, \ldots, m_n), l, t, r, s_u)$		Vérificateur $\mathcal{V}(pk, (B_0, C, E))$
$a_1, a_2, \ldots, a_{n+6} \xleftarrow{R} \mathbb{Z}_p^*$ $t_1 \leftarrow C^{a_1} f^{a_2}$ $t_2 \leftarrow g_1^{a_3} g_2^{a_4} \ldots g_n^{a_{n+2}} g^{a_{n+3}} B_0^{a_{n+4}} f^{a_2}$ $t_3 \leftarrow E^{a_{n+5}} f^{a_{n+6}}$		
$c \leftarrow \mathcal{H}(Ch, t_1, t_2, t_3)$	$\xleftarrow{Ch}$	$Ch \xleftarrow{R} \mathbb{Z}_p^*$
$R_1 \leftarrow a_1 + c/l, R_2 \leftarrow a_2 + ct$	$\xrightarrow{c, R_1, \ldots, R_{n+6}}$	$t_1' \leftarrow C^{R_1} f^{R_2} E^{-c}$
pour $i \in \{1, \ldots, n\}$, $R_{i+2} \leftarrow a_{i+2} + cm_i$		$t_2' \leftarrow g_1^{R_3} \ldots g_n^{R_{n+2}} g^{R_{n+3}} B_0^{R_{n+4}} f^{R_2} H^{-c}$
$R_{n+3} \leftarrow a_{n+3} + cs_u, R_{n+4} \leftarrow a_{n+4} - cr/l$		$t_3' \leftarrow E^{R_{n+5}} f^{R_{n+6}} C^{-c}$
$R_{n+5} \leftarrow a_{n+5} + cl, R_{n+6} \leftarrow a_{n+6} - ctl$		Tester si $c = \mathcal{H}(Ch, t_1', t_2', t_3')$

FIGURE 4.3 – Détail de la preuve π_3 de présentation de l'accréditation

Nous allons prouver que π_3 est bien une preuve de connaissance qui respecte donc les propriétés de complétude, de validité et de divulgation nulle de connaissance.

Soit $\mathcal{H}$ une fonction de hachage (voir Définition 12), connues des deux parties.

Complétude. Immédiate par simple vérification.

Validité. La *validité* découle de la propriété d'extraction de la preuve de connaissance. Cette propriété implique que pour tout prouveur $\mathcal{P}^*$ qui convainc le vérificateur $\mathcal{V}$ avec une probabilité ε, il existe un extracteur qui interagit avec $\mathcal{P}^*$ et retourne $(\alpha, \beta, \lambda, \delta_1, \ldots, \delta_{n+1}, \gamma, \theta)$ avec une probabilité polynômiale en ε. Ainsi, si nous supposons que

l'extracteur possède en entrée deux transcriptions distinctes de cette même preuve, à savoir $(p, \mathbb{G}, g, h, f, B_0, C, E, c, \tilde{c}, R_1, \ldots, R_{n+6}, \tilde{R}_1, \ldots, \tilde{R}_{n+6})$ avec $c \neq \tilde{c}$. Si $(c - \tilde{c})$ est inversible dans $\mathbb{Z}_p$ alors les valeurs peuvent être obtenues à partir des relations suivantes, modulo p :

$$\alpha = \frac{R_1 - \tilde{R}_1}{c - \tilde{c}}; \beta = \frac{R_2 - \tilde{R}_2}{c - \tilde{c}}; \text{ pour tout } i \text{ dans } \{1, \ldots, n\}, \delta_i = \frac{R_{i+2} - \tilde{R}_{i+2}}{c - \tilde{c}};$$

$$\delta_{n+1} = \frac{R_{n+3} - \tilde{R}_{n+3}}{c - \tilde{c}}; \lambda = \frac{R_{n+4} - \tilde{R}_{n+4}}{c - \tilde{c}}; \theta = \frac{R_{n+5} - \tilde{R}_{n+5}}{c - \tilde{c}}, \gamma = \frac{R_{n+6} - \tilde{R}_{n+6}}{c - \tilde{c}}.$$

Nous savons que $H = E \cdot h^{-1} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta}$, d'où $E = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta} h$. En outre, étant donné que $E = C^{\alpha} f^{\beta}$, nous avons les relations suivantes :

$$\begin{aligned} & C^{\alpha} f^{\beta} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta} h \\ \Rightarrow \quad & C^{\alpha} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} h \qquad (4.1) \\ \Rightarrow \quad & B_0^{\alpha y} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} h \\ \Rightarrow \quad & B_0^{\alpha y - \lambda} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} h \qquad (4.2) \end{aligned}$$

Si α est différent de 0 alors l'équation (4.2) implique que

$$(B_0^{\alpha})^{y - \frac{\lambda}{\alpha}} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} h. \qquad (4.3)$$

Nous posons alors $A = B_0^{\alpha}$, $r = -\frac{\lambda}{\alpha}$, $s_u = \delta_{n+1}$ et $m_i = \delta_i$ for $i \in \{1, \ldots, n\}$. Pour $\alpha \neq 0$, l'équation (4.3) implique que le prouveur connaît un $\mathsf{MAC}_{\mathsf{BB}}^n$ valide (A, r, s_u) sur les messages $(m_1, \ldots, m_n)$. Notons que $y - \frac{\lambda}{\alpha}$ est différent de 0 car, dans le cas contraire, cela impliquerait que le prouveur connaît la clé secrète y qui serait alors égale à $\frac{\lambda}{\alpha}$.

Nous allons prouver que α est bien différent de 0 en partant du résultat suivant :

$$\begin{aligned} & C = E^{\theta} f^{\gamma} = (C^{\alpha} f^{\beta})^{\theta} f^{\gamma} = C^{\alpha\theta} f^{\beta\theta + \gamma} \\ \Rightarrow \quad & 1 = C^{\alpha\theta - 1} f^{\beta\theta + \gamma} \qquad (4.4) \end{aligned}$$

> Si le prouveur ne connaît pas le logarithme discret de C en base f alors il ne connaît qu'une unique représentation $(0, 0)$ de 1 dans la base (C, f) [Bra93]. D'où, $\alpha\theta = 1$ et donc $\alpha \neq 0$.

> Supposons maintenant que $\alpha = 0$ et que le prouveur connaisse le logarithme discret de C dans la base f, noté $\chi : C = f^{\chi}$. À partir de $C = B_0^y$, nous avons $B_0^y = f^{\chi}$ et donc $B_0 = f^{\frac{\chi}{y}}$ – car $Y = g_0^y \neq 1$ implique que $y \neq 0 \mod p$. Étant donné l'équation (4.1) et que α est supposé égal à 0, nous avons :

$$h = g_1^{-\delta_1} g_2^{-\delta_2} \ldots g_n^{-\delta_n} g^{-\delta_{n+1}} f^{-\lambda\frac{\chi}{y}}.$$

Ainsi, l'émetteur pourrait utiliser le prouveur comme sous-routine pour calculer une représentation de h dans la base $(g_1, g_2, \ldots, g_n, g, f)$. Étant donné que $(g_1, g_2, \ldots, g_n, g, f)$ est un ensemble de générateurs aléatoires de $\mathbb{G}$, être en mesure de calculer une telle

représentation de h contredirait l'hypothèse du logarithme discret DL [Bra93]. Cela implique que soit le prouveur $\mathcal{P}^*$ ne connaît pas le logarithme discret de C dans la base f, soit que α est différent de 0. Dans tous les cas, cela signifie que α est bien différent de 0 et que le prouveur connaît donc un $\mathsf{MAC}^n_{\mathsf{BB}}$ valide (A, r, s_u) sur les messages $(m_1, \ldots, m_n)$.

Non-divulgation (vérificateur honnête). Nous construisons un simulateur Sim qui simule les échanges avec tout vérificateur honnête $\mathcal{V}^*$ comme suit :

1. Sim choisit aléatoirement l' dans $\mathbb{Z}_p^*$ et un générateur E dans $\mathbb{G}$ puis calcule :

$$B_0 = {g_0}^{l'} \quad \text{et} \quad C = Y^{l'}.$$

2. Sim choisit aléatoirement $c, R_1, \ldots, R_{n+6}$ dans $\mathbb{Z}_p^*$ et génère :

$$t_1 = C^{R_1} f^{R_2} E^{-c}, \quad t_2 = g_1^{R_3} \ldots g_n^{R_{n+2}} g^{R_{n+3}} B_0^{R_{n+4}} f^{R_2} (E \cdot h^{-1})^{-c} \quad \text{et} \quad t_3 = E^{R_{n+5}} f^{R_{n+6}} C^{-c}.$$

3. Sim retourne alors $S = \{B_0, C, E, c, R_1, R_2, \ldots, R_{n+6}\}$.

Étant donné que $\mathbb{G}$ est d'ordre premier, la première étape ne révèle aucune information. En effet, il existe un x dans $\mathbb{Z}_p$ tel que, pour un $\mathsf{MAC}^n_{\mathsf{BB}}$ valide (A, r, s_u) sur $(m_1, \ldots, m_n)$, nous avons $A = g_0^x$. Il existe l dans $\mathbb{Z}_p^*$ tel que $\mathbb{Z}_p^*$ tel que $B_0 = A^l = g_0^{lx} = g_0^{l'}$ pour $l' = lx$. Ceci implique que $C = A^{ly} = Y^{l'}$. En outre, il existe un t tel que $E = C^{\frac{1}{l}} f^t$. Ainsi, la sortie S du simulateur et la vue réelle du vérificateur $\mathcal{V}^*$ au cours du protocole sont statistiquement indistinguables.

Théorème 8. *Notre schéma d'accréditations anonymes avec vérification par clé est* résistant aux falsifications *sous l'hypothèse que* $\mathsf{MAC}^n_{\mathsf{BB}}$ *est sUF-CMVA et vérifie les propriétés d'*anonymat parfait*, d'*émission aveugle *et de* consistance des paramètres *dans le modèle de l'oracle aléatoire.*

Démonstration. La preuve du Théorème 8 est disponible à la Section 4.2.3. □

Vers un schéma à clé publique. Ce schéma de KVAC peut être adapté pour obtenir un schéma d'accréditations anonymes classique, où la vérification est réalisée avec la clé publique. Ainsi, un utilisateur peut prouver la possession d'une accréditation auprès de toute entité, même si cette dernière ne connaît pas la clé privée. Pour cela, nous utilisons un environnement bilinéaire (voir Définition 7). Les étapes précédentes sont mises à jour comme suit :

Initialisation. Les paramètres publics pp deviennent $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_0, g_1, \ldots, g_n, g, h, f, \tilde{g}_0, e)$ où $\mathbb{G}_1, \mathbb{G}_2$ et $\mathbb{G}_T$ sont trois groupes cycliques d'ordre premier p, $(g_0, \{g_i\}_{i=1}^n, g, h, f)$ sont des générateurs aléatoires de $\mathbb{G}_1$, $\tilde{g}_0$ est un générateur aléatoire de $\mathbb{G}_2$ et $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ est une application bilinéaire.

Génération des clés. Une seconde clé publique $W = \tilde{g}_0^y$ de l'émetteur est construite, associée à la clé privée y. Ici, la clé privée n'est plus partagée avec le vérificateur, elle n'est connue que de l'émetteur.

Accréditation anonyme. Le protocole d'émission de l'accréditation anonyme, exécuté entre $\mathcal{U}$ et l'émetteur $\mathcal{E}$, reste identique.

Présentation de l'accréditation. Le vérificateur ne connaît pas la clé privée y. Il utilise donc le couplage e pour tester la validité de l'accréditation sur $(m_1, \ldots, m_n)$. Ainsi, à partir des valeurs B_0, C et E envoyées par $\mathcal{U}$ et des paramètres publiques W et $\tilde{g}_0$, $\mathcal{V}$ accepte l'accréditation si la relation suivante est satisfaite :

$$e(C, \tilde{g}_0) = e(B_0, W)$$

et que la preuve π_3 est correcte.

4.2.2 Analyse de performance et de sécurité

Dans cette section, nous étudions les performances atteintes et la sécurité garantie par notre schéma d'accréditations anonymes avec vérification par clé. Pour cela, nous commençons par comparer notre système aux principales constructions mises en avant dans la partie état de l'art et nous présentons les résultats d'une implémentation réalisée sur une carte SIM NFC (*Near Field Communication*). Enfin, nous prouvons la sécurité de notre schéma à travers des preuves de sécurité pour chacune des propriétés attendues.

Comparaison avec l'existant

Le protocole `Show`, où l'accréditation est utilisée, est l'étape la plus coûteuse d'un schéma d'accréditations anonymes. Pour comparer le nôtre aux schémas les plus répandus – à savoir les systèmes *U-Prove*, *Idemix*, CL bilinéaire et ceux construits à partir de $\mathsf{MAC}_{\mathsf{GGM}}$ ou $\mathsf{MAC}_{\mathsf{DDH}}$ – nous étudions la taille des accréditations, le coût de création de la preuve de présentation de l'accréditation et la complexité en nombre d'éléments de groupe.

Schémas	Taille (en bits) de l'accréditation	Nombre d'exponentiations
U-Prove	$1024s$	$2c$ 2-exp et 1 $(n-r+1)$-exp
Idemix	5369	1 1-exp(2048), c 2-exp(256,2046), c 2-exp(592,2385) et 1 $(n-r+2)$-exp (456,3060,592,...,592)
CL bilinéaire	$512n$+768	$(3+n)$ 1-exp, $2c$ 2-exp et $3+n$ couplages
$\mathsf{MAC}_{\mathsf{GGM}}$	512	3 1-exp, $2(n-r)$ 2-exp et 1 $(n-r+1)$-exp
$\mathsf{MAC}_{\mathsf{DDH}}$	1024	6 1-exp, $2(n-r+1)$ 2-exp et 2 $(n-r+1)$-exp
$\mathbf{MAC}^{n}_{\mathbf{BB}}$	**1024**	**1 1-exp, 4 2-exp and 1 $(n-r+3)$-exp**

TABLE 4.1 – Comparaison des tailles d'accréditation, pour s utilisations non-chaînables, et du coût de la preuve de présentation de l'accréditation, pour n attributs dont c restent cachés. Tous ces schémas utilisent une courbe elliptique de 256 bits, excepté *Idemix* qui utilise un module RSA de 2048 bits.

Coût de la preuve de présentation de l'accréditation. Pour les différents schémas, le Tableau 4.1 permet de comparer le coût de construction de la preuve de présentation de l'accréditation en nombre total de multi-exponentiations. Nous utilisons la même notation que celle utilisée dans [CMZ14] : l-exp désigne le calcul d'un produit de l puissances et l-exp$(b_1, \ldots, b_l)$ correspond au calcul de l puissances avec des exposants de tailles $b_1, \ldots, b_l$ bits, dans le cas

d'*Idemix*. Le nombre de multi-exponentiations dépend de trois paramètres : le nombre n d'attributs considérés dans l'accréditation, le nombre r d'attributs révélés et le nombre c d'attributs qui restent secrets.

Notre schéma basé sur $\mathsf{MAC}^n_{\mathsf{BB}}$ concurrence *U-Prove* qui n'assure pas la non-chaînabilité multiple et le système basé sur $\mathsf{MAC}_{\mathsf{GGM}}$ qui nécessite obligatoirement que le vérificateur connaisse la clé privée de l'émetteur. Lorsque la plupart des attributs restent secrets, notre schéma se révèle plus performant que celui utilisant $\mathsf{MAC}_{\mathsf{GGM}}$.

Complexité en nombre d'éléments du groupe. Étant donné qu'une unique mise en gage suffit pour tous les attributs non révélés, notre preuve de présentation de l'accréditation a une complexité en $O(1)$ en nombre d'éléments du groupe. Ainsi, notre schéma d'accréditation anonyme est plus efficace que celui de CHASE, MEIKLEJOHN et ZAVERUCHA [CMZ14] – basé sur $\mathsf{MAC}_{\mathsf{GGM}}$ ou sur $\mathsf{MAC}_{\mathsf{DDH}}$ – qui a une complexité en $O(c)$ puisque c mises en gage sont nécessaires, une pour chaque attribut non-révélé.

Implémentation

Pour montrer l'efficacité de notre schéma, nous avons également réalisé une implémentation du protocole `Show` et donc de la preuve de présentation de l'accréditation, sur une carte SIM NFC classique.

Environnement. La partie "utilisateur" est implémentée sur une carte SIM Javacard 2.2.2, conforme à GlobalPlatform, et embarquée dans un téléphone NFC Samsumg Galaxy S3. Comparée aux spécifications Javacard, la seule particularité de notre carte est la présence d'API (*Application Programming Interface*) additionnelles, fournies par l'encarteur Oberthur, qui permettent de réaliser des opérations modulaires et de l'arithmétique sur courbes elliptiques. Pour supporter ces calculs de cryptographie à clé publique, basée sur les courbes elliptiques, la carte SIM est munie d'un crypto-processeur. Elle est donc plus performante que la plupart des cartes déployées aujourd'hui. Cependant, il est important de noter que ce type de carte SIM est déjà largement déployé par certains opérateurs tels qu'Orange ou SFR qui offrent des services basés sur le NFC.

La partie "vérificateur" est exécutée sur un PC Quad-Core Intel Xeon CPU @3.70GHz. Les communications entre la carte SIM, le téléphone et l'ordinateur sont réalisées par NFC avec un lecteur PC/SC standard, un Omnikey 5321.

Étude des résultats. L'implémentation utilise une courbe elliptique Barreto-Naehrig [BN06] sur un corps de 256 bits, adaptée à l'utilisation de couplages. Le protocole est divisé en deux parties :

- une partie "hors ligne" qui peut être exécutée à l'avance par la carte et qui permet de calculer toutes les valeurs nécessaires à l'exécution de `Show` dans le pire cas à savoir lorsqu'aucun attribut n'est révélé explicitement ;
- une partie "en ligne" qui ne peut être réalisée que lors de l'exécution de `Show` puisqu'elle dépend du challenge envoyé par le vérificateur pour la preuve π_3.

En effet, dans notre implémentation, la preuve π_3 est non-interactive : le vérificateur envoie donc un challenge *Ch* à l'utilisateur qui est utilisé ensuite par ce dernier pour générer son challenge c avec une fonction de hachage.

Le Tableau 4.2 représente les temps d'exécution obtenus pour cet environnement pour $n = 3$ attributs considérés dont $c = 1$ caché et $r = n - c = 2$ révélés.

Hors Ligne (carte) - Batterie On : (1352 - 1392) 1378ms	
En Ligne	
Construction de la preuve (carte)	*Vérification de la preuve (PC)*
Batterie On : (81-86) 83 ms Batterie Off : (123-124) 123.4 ms	cas y connu : (3-14) 5 ms cas y inconnu : (5-17) 10 ms
Total En Ligne	
Batterie On	cas y connu : (84-100) 88 ms cas y inconnu : (86-103) 93 ms
Batterie Off	cas y connu : (126-137) 128 ms cas y inconnu : (128-141) 133 ms

TABLE 4.2 – Temps d'exécution ((min-max) moyen) en ms du protocole `Show`

Le terme "Batterie Off" désigne un téléphone éteint soit volontairement par l'utilisateur, soit parce que la batterie est déchargée. Dans ce cas, les standards NFC assurent que l'accès NFC à la carte SIM est toujours possible mais avec des performances amoindries. Pour les calculs réalisés hors ligne, nous supposons qu'ils sont initialisés lorsque le téléphone est allumé, "Batterie On", et automatiquement après chaque preuve de présentation afin d'anticiper la prochaine. Les calculs en ligne correspondent aux calculs des réponses R_i et du haché c' utilisés pour la preuve π_3. Ils peuvent être réalisés même dans le cas d'un téléphone éteint. Il est important de préciser que tous les calculs sont entièrement réalisés par la carte. En particulier, le téléphone est uniquement utilisé pour déclencher le protocole `Show` et pour alimenter la carte.

En moyenne, la partie "en ligne" de la preuve est très rapide, y compris quand le téléphone est éteint. Finalement, ce sont les communications qui représentent la phase la plus coûteuse.

4.2.3 Preuves de sécurité

Nous prouvons ici le Théorème 8 relatif à la sécurité de notre schéma d'accréditations anonymes avec vérification par clé. Nous nous basons sur le modèle de sécurité défini à la Section 4.1.2 et présentons chacune des preuves de sécurité séparément.

Preuve de complétude

Deux propriétés différentes doivent être démontrées. La première est immédiate et découle de la propriété de complétude de notre $\mathsf{MAC}^n_{\mathsf{BB}}$. Nous détaillons ici la deuxième. L'algorithme `Issue`$(sk, (m_1, \ldots, m_n))$ génère des accréditations de la forme

$$A^{y+r} = g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^{s_u} h.$$

Ensuite, si l'algorithme `Show` est exécuté honnêtement par les deux parties, π_3 est acceptée grâce à la propriété de complétude propre aux preuves de connaissance. Les valeurs suivantes

sont également calculées, pour l aléatoire dans $\mathbb{Z}_p^*$:

$$\begin{aligned} B_0 &= A^l \quad \text{et,} \\ C &= \tilde{C}_{\vec{m}}^{l} \cdot B_0^{-r} \\ &= g_1^{lm_1} \dots g_n^{lm_n} g^{ls} h^l \cdot B_0^{-r} \\ &= (g_1^{m_1} \dots g_n^{m_n} g^s h)^l \cdot B_0^{-r} \\ &= (A^{y+r})^l \cdot B_0^{-r} \\ &= (A^l)^y (A^l)^r \cdot B_0^{-r} \quad = \quad B_0^y B_0^r B_0^{-r} \quad = \quad B_0^y \end{aligned}$$

Par conséquent, la vérification $C = B_0^y$ réalisée par $\mathcal{V}$ fonctionne et l'accréditation sera acceptée.

Preuve de résistance aux falsifications

Soit $\mathcal{A}$ un adversaire contre la propriété de résistance aux falsifications de notre schéma, pour les accréditations générées avec `BlindIssue`. Il a accès à deux oracles $\mathcal{O}_{\texttt{BlindIssue}}$ et $\mathcal{O}_{\texttt{ShowVerif}}$. Pour une mise en gage $C_{\vec{m}}$ et une preuve π_1, $\mathcal{O}_{\texttt{BlindIssue}}$ retourne une accréditation valide $\sigma = (A, r, s')$ et une preuve π_2. Pour des valeurs (B_0, C, E) et une preuve π_3, l'oracle $\mathcal{O}_{\texttt{ShowVerif}}$ teste la validité de l'accréditation et retourne 1 si elle est acceptée, 0 sinon.

La réduction $\mathcal{R}$ utilise l'adversaire $\mathcal{A}$ contre la propriété sUF-CMVA de notre schéma $\mathsf{MAC}_{\mathsf{BB}}^n$ dont la preuve de sécurité est disponible à la Section 3.1.2. En entrée, $\mathcal{R}$ utilise les paramètres publics pp fournis par le challenger $\mathcal{C}$ et Y, valeur publique de l'émetteur. Cette réduction a accès aux deux oracles $\mathcal{O}_{\texttt{MAC}_{\texttt{BB}}^n}$ et $\mathcal{O}_{\texttt{Verif}_{\texttt{BB}}^n}$ qui respectivement génère et vérifie un $\mathsf{MAC}_{\mathsf{BB}}^n$. $\mathcal{R}$ fournit l'ensemble des paramètres publics (pp, Y) à $\mathcal{A}$ et répond à ses requêtes comme suit, à l'aide de ses oracles :

- Requêtes $\mathcal{O}_{\texttt{BlindIssue}}$: étant donné $C_{\vec{m}}$ et π_1, $\mathcal{R}$ retourne $\perp$ si π_1 est fausse et procède au calcul d'une accréditation sinon. Dans ce cas, $\mathcal{R}$ exécute l'extracteur de la preuve π_1 pour obtenir $(m_1, \dots, m_n, s)$ puis appelle son oracle $\mathcal{O}_{\texttt{MAC}_{\texttt{BB}}^n}$ sur $(m_1, \dots, m_n)$ qui lui retourne (A, r, s_u). Pour finir, $\mathcal{R}$ simule la preuve π_2 et transmet $((A, r, s_u - s), \pi_2)$ à $\mathcal{A}$.
- Requêtes $\mathcal{O}_{\texttt{ShowVerif}}$: étant donné (B_0, C, E) et π_3, $\mathcal{R}$ retourne $\perp$ si π_3 n'est pas valide et poursuit la vérification comme suit sinon. $\mathcal{R}$ utilise l'extracteur de la preuve de connaissance pour obtenir les valeurs α, β, γ, δ_1, δ_2, …, δ_{n+1}, γ et θ. Dans le cas où α est nul, $\mathcal{R}$ retourne 0 à $\mathcal{A}$. Sinon, $\mathcal{R}$ calcule $A = B_0^{\alpha}$, $r = -\lambda/\alpha$ et $s = \delta_{n+1}$. Enfin, la réduction fait appel à l'oracle $\mathcal{O}_{\texttt{Verif}_{\texttt{BB}}^n}$ avec $((\delta_1, \delta_2, \dots, \delta_n), (A, r, s))$ et retourne le résultat à $\mathcal{A}$.

À la fin du protocole `Show`, la réduction $\mathcal{R}$ extrait à nouveau α, β, γ, δ_1, δ_2, …, δ_{n+1}, γ et θ et retourne alors sa contre-façon $\big((\delta_1, \delta_2, \dots, \delta_n), (B_0^{\alpha}, -\frac{\lambda}{\alpha}, \delta_{n+1})\big)$.

Notons que les réponses données par la réduction $\mathcal{R}$ aux requêtes $\mathcal{O}_{\texttt{BlindIssue}}$ sont identiques à celles du vrai protocole `BlindIssue`. Par conséquent, nous considérons que les réponses aux requêtes $\mathcal{O}_{\texttt{ShowVerif}}$ sont, avec une probabilité écrasante, identiques à celles du véritable algorithme `ShowVerif`. En effet, par ses propriétés, la preuve de connaissance π_3 garantit que l'extracteur réussit à produire des témoins valides avec une probabilité écrasante. De plus, s'il fournit en sortie (α, β, γ, δ_1, δ_2, …, δ_{n+1}), étant donné que $E = C^{\alpha} f^{\beta} =$

$g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \cdot B_0^{\lambda} \cdot f^{\beta}$, nous avons :

$$C^{\alpha} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \cdot B_0^{\lambda}$$
$$\text{d'où} \quad C^{\alpha} \cdot B_0^{-\lambda} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h$$

Si l'oracle $\mathcal{O}_{\mathtt{Verif}^n_{\mathsf{BB}}}$ retourne 1 pour l'entrée $\big((\delta_1, \delta_2, \dots, \delta_n), (B_0^{\alpha}, -\frac{\lambda}{\alpha}, \delta_{n+1})\big)$ alors

$$\begin{aligned}
& (B_0^{\alpha})^{y-\frac{\lambda}{\alpha}} && = && g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \\
\Rightarrow \quad & (B_0^{\alpha})^{y} \cdot B_0^{-\lambda} && = && g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \\
\Rightarrow \quad & (B_0^{\alpha})^{y} \cdot B_0^{-\lambda} && = && C^{\alpha} B_0^{-\lambda} \\
\Rightarrow \quad & (B_0^{\alpha})^{y} && = && C^{\alpha} \\
\Rightarrow \quad & B_0^{y} && = && C
\end{aligned}$$

Par ailleurs, la valeur α est nécessairement différente de 0 puisque dans le cas contraire B_0^{α} aurait été égal à 1 et donc rejeté par l'oracle.

Ainsi, l'algorithme honnête de vérification accepte si et seulement si $(B_0^{\alpha}, -\frac{\lambda}{\alpha}, \delta_{n+1})$ est lui-même accepté par l'algorithme de vérification de $\mathsf{MAC}^n_{\mathsf{BB}}$ pour la séquence de messages $(\delta_1, \delta_2, \dots, \delta_n)$.

De la même façon, nous considérons que la réduction $\mathcal{R}$ peut extraire un MAC valide à partir du protocole `Show` avec $\alpha \neq 0$ et la vérification `ShowVerif` retournera 1. Ainsi, si l'adversaire $\mathcal{A}$ peut faire accepter un prédicat ϕ qui n'est pas réellement satisfait par l'un des attributs de l'ensemble envoyé à l'oracle $\mathcal{O}_{\mathtt{BlindIssue}}$ alors $\mathcal{R}$ peut extraire une séquence de messages $(\delta_1, \delta_2, \dots, \delta_n)$ et un MAC valide pour cette séquence. Cela contredit la propriété de résistance aux falsifications du schéma $\mathsf{MAC}^n_{\mathsf{BB}}$.

Preuve d'anonymat parfait

Nous supposons que l'utilisateur essaye de prouver qu'il détient une accréditation pour des attributs satisfaisant un prédicat ϕ. Notre but est de montrer qu'il existe un algorithme efficace `SimShow` qui, pour tout adversaire $\mathcal{A}$, est indistinguable du protocole `Show` mais qui ne prend en entrée que ϕ et la clé secrète y. Nous considérons ϕ dans Φ et $(m_1, \dots, m_n)$ dans Att tel que $\phi(m_1, \dots, m_n) = 1$. En outre, nous supposons que l'ensemble des paramètres publics pp du système, la valeur publique Y de l'émetteur et σ vérifient $\mathtt{CredVerif}(y, \sigma, (m_1, \dots, m_n)) = 1$. Ainsi, $\sigma = (A, r, s_u)$ dans $\mathbb{G} \times \mathbb{Z}_p \times \mathbb{Z}_p$ sur $\tilde{C}_{\vec{m}}$ dans $\mathbb{G}$ satisfait la relation suivante : $A^{y+r} = g_1^{m_1} \dots g_n^{m_n} g^{s_u} h$.

L'algorithme `SimShow` fonctionne comme suit. Après avoir choisi aléatoirement l' dans $\mathbb{Z}_p^*$ et un générateur E dans $\mathbb{G}$, il calcule $B_0 = g_0^{l'}$ et $C = Y^{l'}$. Il exécute ensuite $\mathcal{A}$ avec les valeurs (B_0, C, E) comme premier message, simule la preuve de connaissance π_3 et retourne ce que $\mathcal{A}$ obtient à la fin de la preuve.

Étant donné que A est différent de 1, il existe un x dans $\mathbb{Z}_p$ tel que $A = g_0^x$. Il existe également une valeur aléatoire l dans $\mathbb{Z}_p$ telle que $B_0 = A^l = g_0^{lx} = g_0^{l'}$ pour $l' = lx$. De plus, $C = A^{ly} = Y^{l'}$ et, il existe un t tel que $E = C^{1/l} f^t$. Ainsi, toutes les valeurs calculées par `SimShow` sont identiques à celles obtenues par l'exécution du protocole `Show`. La propriété de

"non-divulgation" de la preuve de connaissance nous permet de conclure qu'une instance de `SimShow` est indistinguable de celle obtenue par l'adversaire lorsqu'il interagit avec `Show`.

Preuve d'émission aveugle

Attributs connus. Comme nous l'avons défini à la Section 4.1.2, nous commençons par le cas où tous les attributs sont connus de l'émetteur qui génère l'accréditation.

(1) Dans le cas où l'utilisateur est corrompu, notre simulateur Sim, sur l'entrée partagée (S, pp, pk), reçoit la séquence $(m_1, \ldots, m_n)$ de l'utilisateur et transfert ces valeurs à la fonctionnalité f. Si S ne correspond pas à $(m_1, \ldots, m_n)$, la fonctionnalité retourne "erreur sur les attributs" et σ dans les autres cas. Si aucune erreur n'est constatée, Sim envoie alors σ et exécute le simulateur associé à π_2 pour obtenir une preuve qui atteste que le calcul de σ est correct. La propriété de "non-divulgation" des preuves de connaissance assure que cette simulation est indistinguable d'une exécution dans le monde réel.

(2) Dans le cas où l'émetteur est corrompu, notre simulateur Sim reçoit en entrée $\sigma = (A, r, s')$ par l'émetteur et exécute la vérification de la preuve π_2. Si π_2 est acceptée, il utilise l'extracteur correspondant à la preuve de connaissance et obtient y, r et s'. Il envoie (y, r, s') à la fonctionnalité. Par les propriétés d'une preuve de connaissance, l'accréditation envoyée est $\sigma = ((g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^{s+s'} h)^{1/(y+r)}, r, s')$; elle correspond exactement à ce que la fonctionnalité aurait produit pour l'entrée (y, r, s').

Attributs cachés. Nous considérons maintenant le cas du protocole `BlindIssue` où les attributs sont cachés à l'émetteur. Nous définissons la fonctionnalité idéale $\mathcal{F}$ qui à partir des entrées (y, r, s') de l'émetteur et $(m_1, \ldots, m_n, s, l)$ de l'utilisateur, retourne à $\mathcal{U}$ une version randomisée σ_l de l'accréditation égale à (A^l, s', r) pour l aléatoire dans $\mathbb{Z}_p^*$; l'émetteur ne reçoit rien.

(1) Dans le cas où l'utilisateur est corrompu, notre simulateur Sim, sur l'entrée partagée (S, pp, pk), reçoit la mise en gage $C_{\vec{m}}$ et exécute la vérification de la preuve de connaissance π_1. Si π_1 est valide, il utilise l'extracteur correspondant et obtient $(m_1, \ldots, m_n)$ et s. Il envoie ces valeurs, avec l, à la fonctionnalité qui retourne $\sigma_l = (A^l, r, s')$. Sim envoie alors σ et exécute le simulateur associé à π_2 pour obtenir une preuve qui atteste que le calcul de σ est correct. La propriété de "non-divulgation" des preuves de connaissance assure que cette simulation est indistinguable d'une exécution dans le monde réel.

(2) Dans le cas où l'émetteur est corrompu, notre simulateur Sim, sur l'entrée partagée (S, pp, pk), reçoit $\sigma = (A, r, s')$, $C_{\vec{m}}$ et π_2 par l'émetteur et exécute la vérification de la preuve π_2. Si π_2 est valide, il utilise l'extracteur correspondant et obtient y, r et s'. Il envoie (y, r, s') à la fonctionnalité. Par les propriétés d'une preuve de connaissance, l'accréditation envoyée est $\sigma = ((C_{\vec{m}} \cdot g^{s'} \cdot h)^{1/(y+r)}, r, s')$; elle est indistinguable de la sortie que la fonctionnalité aurait produite pour l'entrée (y, r, s').

Preuve de consistance des paramètres

Notre schéma d'accréditation anonyme avec vérification par clé satisfait cette propriété de façon immédiate. En effet, les paramètres publics du système pp contiennent un groupe cyclique $\mathbb{G}$ d'ordre premier p et $(n+4)$ générateurs $g_0, g_1, \ldots, g_n, g, h, f$. La valeur publique Y est dans $\mathbb{G}$; elle est associée à la clé secrète y de l'émetteur $\mathcal{E}$ qui correspond au logarithme

discret de Y en base g_0. Or, étant donné que le logarithme discret est unique, un adversaire ne peut pas trouver deux clés secrètes différentes correspondant à la même valeur publique Y.

Conclusion

Dans ce chapitre, nous avons présenté en détail un nouveau schéma d'accréditations anonymes avec vérification par clé, efficace et sûr sous l'hypothèse que $\mathsf{MAC}^n_{\mathsf{BB}}$ est résistant aux falsifications. Par ailleurs, il satisfait la non-chaînabilité multiple et est adapté à l'usage sur des périphériques aux capacités limitées puisqu'aucun couplage n'est nécessaire du côté du prouveur. Le protocole de présentation d'une accréditation a une complexité en $O(1)$, en nombre d'éléments de groupe ; une clé secrète unique suffit quelque soit le nombre d'attributs non-révélés considérés. En outre, il peut être adapté pour devenir publiquement vérifiable sans ajouter de calculs pour l'utilisateur.

Dans le chapitre suivant, nous détaillons un système de vote électronique basé sur notre MAC séquentiellement agrégé. Il assure la propriété dite de résistance à la coercition et les accréditations peuvent être simplement mises à jour, pour être utilisées lors d'élections successives, ou révoquées.

Chapitre 5

Conception d'un système de vote électronique

Dans ce chapitre, nous introduisons un nouveau schéma de vote électronique efficace et résistant à la coercition. Sa complexité linéaire en nombre de bulletins le rend utilisable pour de réelles élections. Les accréditations anonymes, nécessaires pour voter, sont générées selon le MAC algébrique $\mathsf{MAC}_{\mathsf{GGM}}$ de CHASE, MEIKLEJOHN et ZAVERUCHA [CMZ14]. Introduit à la Section 3.2.1, notre MAC séquentiellement agrégé permet de mettre à jour efficacement ces accréditations pour qu'elles puissent être encore utilisées lors d'un nouveau scrutin ou pour révoquer un électeur qui n'est plus éligible.

Ce résultat a été publié au *workshop* VOTING'16, associé à la conférence FC 2016, dans l'article *"Remote Electronic Voting can be Efficient, Verifiable and Coercion-Resistance"* [ABBT16] avec notre MAC séquentiellement agrégé.

Sommaire

5.1 Vote électronique

Le vote électronique est un système de vote dématérialisé qui offre de nombreux avantages. Un votant peut ainsi soumettre son bulletin depuis son ordinateur personnel ou encore son téléphone portable. Il n'a plus besoin de se déplacer à son bureau de vote et le fait de voter est ainsi moins contraignant. Cela pourrait attirer plus de votants et, par conséquent, augmenter la participation électorale. En outre, étant donné que les bulletins sont électroniques, le dépouillement par les autorités compétentes pourrait être réalisé plus efficacement et sans risque d'erreurs de comptage.

Certains pays, comme l'Estonie ou la Suisse par l'exemple, ont déjà choisi d'utiliser le vote électronique pour des élections politiques. Cependant, ce type de scrutin n'est pas encore très répandu. En effet, bien qu'offrant plus de souplesse en matière d'utilisation, un système de vote électronique implique des risques importants en terme de sécurité. Des attaques dites par déni de service pourraient être menées, ce qui empêcheraient les électeurs de voter. Un logiciel malveillant pourrait modifier le choix d'un électeur à son insu. De même, il pourrait tout simplement communiquer le vote à une tierce partie sans que l'électeur ne s'en aperçoive, ce qui engendrerait alors de nouveaux risques [US 15]. Dans nos travaux, nous nous sommes focalisés sur ces possibles menaces comme la pression que pourrait subir un électeur au moment de voter. En effet, en l'absence d'isoloir, l'électeur pourrait être contraint de voter pour un candidat donné ou même être tenté de vendre son vote au plus offrant. Nous supposerons toutefois que le dispositif de vote utilisé par l'électeur – ordinateur ou téléphone portable, par exemple – n'est pas compromis et que les bulletins sont émis comme il le souhaite (*cast-as-intended*).

Dans cette section, nous présentons l'état de l'art principal des systèmes de vote électronique. Nous définissons ensuite formellement le vote électronique et son modèle de sécurité, en particulier la propriété de *résistance à la coercition*.

5.1.1 État de l'art

Pour répondre aux problématiques induites par l'absence d'isoloir, JUELS, CATALANO et JAKOBSSON [JCJ05] introduisent en 2005 la propriété de "résistance à la coercition". Celle-ci considère les différentes attaques qui pourrait avoir lieu contre un votant comme contraindre un électeur à voter pour un candidat spécifique, le forcer à s'abstenir ou encore à révéler son accréditation personnelle afin de voter à sa place. Pour satisfaire cette propriété, les auteurs proposent l'utilisation d'accréditations anonymes. Ainsi, afin d'être en mesure de voter pour une élection donnée, l'électeur doit tout d'abord obtenir une accréditation valide qui est personnelle et privée. En cas de coercition, le votant peut utiliser une fausse accréditation qui sera indistinguable d'une vraie pour l'attaquant. De cette façon, un électeur peut tromper l'attaquant en lui faisant croire qu'il obéit à ses attentes alors que le vote généré sera considéré invalide par les autorités en charge du dépouillement. Cependant, ce premier schéma est inefficace pour des scrutins avec de nombreux électeurs. En effet, pour n bulletins, la complexité de l'étape de dépouillement est quadratique $O(n^2)$.

Pour améliorer ce système, d'autres schémas ont été proposés afin de satisfaire la propriété de résistance à la coercition. Nous présentons ici les plus importants.

En 2007, ARAÙJO, FOULLE et TRAORÉ [AFT08] proposent le premier schéma de vote électronique satisfaisant la propriété de résistance à la coercition, avec une complexité linéaire. Cependant, leur système ne permet pas de participer à plusieurs élections en utilisant la même accréditation. Pour chaque nouveau scrutin, le votant doit se réenregistrer pour obtenir une accréditation dédiée.

Par la suite, ARAÙJO et TRAORÉ [AT13] introduisent en 2013 un schéma qui permet de révoquer les accréditations des votants qui ne sont plus éligibles et de mettre à jour celles des autres votants afin qu'ils puissent prendre part à des scrutins ultérieurs. Les accréditations correspondent à des signatures de groupes BBS [BBS04] qui sont générées de manière conjointe

par les autorités d'enregistrement. Malheureusement, il n'existe pas à ce jour de solution pratique pour calculer ces signatures de façon distribuée. Leur schéma ne serait donc pas utilisable pour des élections réelles.

Ils présentent toutefois une méthode générique pour identifier les accréditations valides mais néanmoins non légitimes qu'une coalition d'autorités d'enregistrement pourrait calculer. Bien que ce risque de collusion soit peu probable, la méthode proposée peut être appliquée à différents systèmes de vote électronique, y compris le nôtre.

Enfin, en 2011, CLARK et HENGARTNER [CH12] ainsi que SPYCHER, KOENIG et HAENNI [SKHS12] proposent deux approches différentes pour satisfaire la propriété de résistance à la coercition. Cependant, leurs schémas n'ont pas réellement une complexité linéaire. En effet, la propriété n'est atteinte qu'en affaiblissant le niveau d'anonymat procuré aux votants. Ainsi, un bulletin n'est plus anonyme par rapport à l'ensemble des votes reçus comme pour le schéma [AFT08] mais seulement par rapport à un sous-ensemble d'entre eux.

5.1.2 Définitions

Un système de vote électronique fait interagir plusieurs participants : un ensemble $\mathcal{E}$ d'autorités d'enregistrement, un ensemble $\mathcal{D}$ d'autorités de dépouillement appelées aussi scrutateurs et un ensemble $\mathcal{V}$ de votants. Par sécurité, le rôle d'autorité d'enregistrement ou de dépouillement est généralement distribué parmi un grand nombre d'entités. Ainsi, toutes ces autorités doivent collaborer pour générer une nouvelle accréditation ou procéder au dépouillement. Individuellement, il leur est impossible de réaliser ces actions.

De façon informelle, les propriétés attendues d'un tel schéma sont les suivantes :
- chaque électeur doit pouvoir s'assurer que son vote a bien été pris en compte ;
- tous les votants doivent être certains que les bulletins comptabilisés sont conformes ;
- seuls les votants légitimes doivent pouvoir participer au scrutin, avec au plus un bulletin ;
- l'électeur doit pouvoir voter pour le candidat de son choix même s'il subit des pressions extérieures et ce sans compromettre le secret de son vote ;
- les votants ne doivent pas être en mesure de prouver pour qui ils ont voté.

Définition 39. Un système de *vote électronique* est constitué des étapes suivantes :

Initialisation. L'initialisation du système est réalisée par l'algorithme `Setup`. Il prend en entrée un paramètre de sécurité λ et retourne les paramètres publics du système.

$$pp \leftarrow \texttt{Setup}(1^\lambda)$$

Génération des clés. L'algorithme de génération des clés `KeyGen` est probabiliste. Il prend en entrée les paramètres publics pp et retourne des paires de clés (clé privée, clé publique) pour les autorités de dépouillement et d'enregistrement. Les autorités de dépouillement reçoivent $(sk_{\mathcal{D}}, pk_{\mathcal{D}})$ et les autorités d'enregistrement reçoivent $(sk_{\mathcal{E}}, pk_{\mathcal{E}})$ qu'elles partagent avec les autorités de dépouillement.

$$\left(\mathcal{E}(sk_{\mathcal{E}}, pk_{\mathcal{E}}), \mathcal{D}\left((sk_{\mathcal{D}}, pk_{\mathcal{D}}), (sk_{\mathcal{E}}, pk_{\mathcal{E}})\right)\right) \leftarrow \texttt{KeyGen}(pp)$$

Enregistrement. Pour s'inscrire à l'élection, le votant $\mathcal{V}_i$ interagit avec les autorités d'enregistrement $\mathcal{E}$ à travers le protocole `Register`. Il fournit son identité ID_i et obtient une

accréditation σ unique, associée à un secret s choisi de manière conjointe par l'ensemble des autorités d'enregistrement $\mathcal{E}$.

$$(\sigma, s) \leftarrow \texttt{Register}(\mathcal{V}_i(pp, ID_i, pk_{\mathcal{E}}), \mathcal{E}(sk_{\mathcal{E}}))$$

Vote. Le votant $\mathcal{V}_i$ réalise son vote avec l'algorithme `Voting`. À partir de la clé publique $pk_{\mathcal{D}}$ des autorités de dépouillement, de l'ensemble O des candidats possibles, de l'accréditation σ et du secret s associé, l'algorithme retourne un bulletin B qui contient le vote v choisi par le votant. Ce bulletin est ajouté à l'ensemble $\mathcal{B}$ des bulletins précédents dans l'urne U.

$$B \leftarrow \texttt{Voting}(pp, pk_{\mathcal{D}}, O, v, (\sigma, s))$$

Pré-Vérification. Les autorités de dépouillement $\mathcal{D}$ réalisent une phase préalable au dépouillement en exécutant l'algorithme `PreVerif`. L'ensemble des bulletins $\mathcal{B}$ présents dans l'urne est mis à jour : les votes invalides sont supprimés, qu'ils aient été mal calculés ou que plusieurs votes soient présents pour un même votant. Ce dernier cas dépend de la politique de sécurité choisie. Par exemple, seul le dernier bulletin posté peut être conservé. L'ensemble des votes restants est noté $\mathcal{B}'$.

$$\mathcal{B}' \leftarrow \texttt{PreVerif}(pp, \mathcal{B}, sk_{\mathcal{E}}, sk_{\mathcal{D}}))$$

Dépouillement. Le dépouillement final de l'élection en cours est réalisé par les autorités de dépouillement $\mathcal{D}$ avec l'algorithme `Tallying`. À partir de l'ensemble $\mathcal{B}'$ des votes restants, tous les votes valides sont ouverts pour obtenir le résultat de l'élection.

$$\mathit{Result} \leftarrow \texttt{Tallying}(pp, \mathcal{B}', sk_{\mathcal{D}})$$

Modèle de sécurité

Outre la propriété classique de complétude, un système de vote électronique est dit sûr s'il satisfait les propriétés de vérifiabilité de bout en bout, d'éligibilité et de résistance à la coercition. Nous donnons ici leurs définitions. Dans le cas de la résistance à la coercition qui est le cœur de notre contribution, nous introduisons l'expérience de sécurité correspondante.

Vérifiabilité de bout en bout. La vérifiabilité de bout-en-bout assure que toutes les étapes du processus de vote peuvent être vérifiées par tous. Elle regroupe différentes propriétés : la vérifiabilité individuelle, universelle et la notion de vote "émis comme prévu".

Vérifiabilité individuelle. Un système de vote électronique assure la propriété de *vérifiabilité individuelle* si le votant peut s'assurer à tout moment que son bulletin est bien présent dans l'urne publique.

Vérifiabilité universelle. Un système de vote électronique assure la propriété de *vérifiabilité universelle* si tous les participants peuvent s'assurer que les conditions suivantes sont respectées :

1. seuls les bulletins valides, publiés par des votants légitimes, sont comptabilisés pour le résultat final ;

2. tous les bulletins valides présents dans l'urne sont pris en compte pour l'élection ; les votes supprimés sont bien invalides.

Enfin, la propriété "*cast-as-intended*" est l'une de nos hypothèses de départ. Elle a pour but d'assurer qu'aucun logiciel malveillant n'est en train de modifier le sens du vote d'un votant à son insu.

Éligibilité. Un système de vote électronique assure la propriété d'éligibilité si tous les participants peuvent s'assurer que les conditions suivantes sont satisfaites :

1. seuls les votants légitimes peuvent soumettre des bulletins de vote ;
2. chaque votant ne peut voter qu'une seule et unique fois.

Résistance à la Coercition. Un système de vote électronique est résistant à la coercition si un votant peut voter selon son propre choix malgré les pressions extérieures qu'il pourrait subir. Autrement dit, le votant est capable de tromper un attaquant en cas de coercition en générant un vote sous la contrainte qui ne sera pas comptabilisé, sans que ce dernier puisse s'en apercevoir. Ainsi, cette propriété garantit également que le votant est incapable de fournir une quelconque information pour prouver qu'il a voté pour un candidat en particulier. Elle est aussi connue sous le nom de *receipt-freeness* c'est-à-dire que le votant ne reçoit aucun reçu exploitable indiquant pour qui il a voté. Par conséquent, il est inutile d'essayer d'intimider un votant ou de vouloir vendre son vote puisqu'aucune preuve ne peut être fournie. L'adversaire ne pourra donc pas déterminer si un votant a voté comme convenu ou non et il ne pourra donc pas influer sur l'élection en cours.

Pour définir formellement la notion de résistance à la coercition, nous reprenons l'expérience de sécurité définie dans l'article [JCJ05]. Nous utilisons alors un algorithme additionnel défini comme suit :

Génération d'une fausse clé. L'algorithme `FakeSecret` prend en entrée la clé publique $pk_{\mathcal{E}}$ des autorités d'enregistrement et la paire (σ, s) appartenant au votant puis retourne un faux secret s'. Pour satisfaire la propriété de résistance à la coercition, ce secret est supposé indistinguable d'un secret valide pour un adversaire $\mathcal{A}$ mais pas pour les autorités de dépouillement.

$$s' \leftarrow \texttt{FakeSecret}\left(pp, pk_{\mathcal{E}}, (\sigma, s)\right)$$

Cette propriété est alors formellement définie par un jeu de sécurité entre un adversaire $\mathcal{A}$ et un votant $\mathcal{V}_j$, ciblé par cet adversaire. Le comportement du votant est déterminé par un bit b tiré aléatoirement au cours de l'expérience. Si b vaut 0 alors le votant essaye de déjouer la tentative de coercition : il publie dans l'urne un vote v pour le candidat β choisi par l'adversaire et lui donne le faux secret s'. Si b vaut 1 alors le votant se prête au jeu de l'adversaire : il ne vote pas mais lui fournit son véritable secret. Le but de l'adversaire est alors de déterminer la valeur du bit b et donc de déduire si $\mathcal{V}_j$ a émis un vote ou non. Bien sûr, l'élection doit être cohérente avec le besoin de résistante à la coercition : nous supposons que l'adversaire n'a pas connaissance des intentions de votes de l'ensemble des participants et que l'élection obéit à une certaine incertitude. En effet, si l'ensemble des participants votent de manière unanime pour un même candidat alors l'attaquant gagne le jeu facilement. La Figure 5.1 détaille cette

$\texttt{Exp}_{\mathcal{A}}^{RC-b}(1^\lambda)$

1. $pp \leftarrow \texttt{Setup}(1^\lambda)$;
2. $\big(\mathcal{E}(sk_{\mathcal{E}}, pk_{\mathcal{E}}), \mathcal{D}\big((sk_{\mathcal{D}}, pk_{\mathcal{D}}), (sk_{\mathcal{E}}, pk_{\mathcal{E}})\big)\big) \leftarrow \texttt{KeyGen}(pp)$;
3. $\mathcal{B} \leftarrow \varnothing$, $\mathcal{CV} \leftarrow \mathcal{A}$(liste des votants éligibles) ;
4. Pour tout votant $\mathcal{V}_i \in \mathcal{V}$, $(\sigma_i, s_i) \leftarrow \texttt{Register}\big(\mathcal{V}_i(pp, ID_i, pk_{\mathcal{E}}), \mathcal{E}(sk_{\mathcal{E}})\big)$;
5. $(j, \beta) \leftarrow \mathcal{A}(\{s_i\}_{i \in \mathcal{CV}})$; // choix de la cible $\mathcal{V}_j$ et du candidat β
6. Si $\mathcal{CV} \neq n_c$ ou si $j \notin \mathcal{V} \backslash \mathcal{CV}$ ou si $\beta \notin \mathcal{O} \cup \varnothing$ alors retourner 0 ;
7. $b \leftarrow \{0, 1\}$;
8. Si $b = 0$ alors // le votant $\mathcal{V}_j$ tente d'éviter la coercition
   $s'_j \leftarrow \texttt{FakeSecret}\big(pp, pk_{\mathcal{E}}, (\sigma, s_j)\big)$;
   $\mathcal{B} \leftarrow \mathcal{B} \cup B_j$ où $B_j \leftarrow \texttt{Voting}(pp, pk_{\mathcal{D}}, \mathcal{O}, \beta, (\sigma_j, s_j))$;
   sinon, $s'_j \leftarrow s_j$;
9. Pour tout votant $\mathcal{V}_i \in \mathcal{V} \backslash \mathcal{CV}$ avec $i \neq j$,
   $\mathcal{B} \leftarrow \mathcal{B} \cup B_i$ où $B_i \leftarrow \texttt{Voting}\big(pp, pk_{\mathcal{D}}, \mathcal{O}, \beta, (\sigma_i, s_i)\big)$;
10. $\mathcal{B} \leftarrow \mathcal{B} \cup \tilde{B}_j \cup \{B_i\}_{i \in \mathcal{CV}}$ où $\tilde{B}_j \leftarrow \mathcal{A}(s'_j, \mathcal{B})$ et $B_i \leftarrow \texttt{Voting}_{\mathcal{A}}\big(pp, pk_{\mathcal{D}}, \mathcal{O}, \beta, (\sigma_i, s_i)\big)$
11. $\mathcal{B}' \leftarrow \texttt{PreVerif}(pp, \mathcal{B}, sk_{\mathcal{E}}, sk_{\mathcal{D}})$
12. $Result \leftarrow \texttt{Tallying}(pp, \mathcal{B}', sk_{\mathcal{D}})$
13. $b' \leftarrow \mathcal{A}(Result)$
14. Retourner $(b' = b)$.

FIGURE 5.1 – Résistance à la coercition

expérience de sécurité $\texttt{Exp}_{\mathcal{A}}^{RC-b}(1^\lambda)$. L'ensemble des votants corrompus par l'adversaire est noté $\mathcal{CV}$. Ainsi, le nombre de votants honnêtes et pour lesquels le vote est incertain est de $(|\mathcal{V}| - |\mathcal{CV}| - 1)$ où le dernier correspond au bulletin émis par le votant sous la contrainte. Un vote peut être nul et est alors noté $\varnothing$.

Pour quantifier l'avantage $\texttt{Adv}_{\mathcal{A}}^{RC}$ de $\mathcal{A}$, nous le comparons avec un adversaire $\mathcal{A}'$ moins puissant qui n'a accès qu'à une version idéale du vote dont l'expérience $\texttt{Exp}_{\mathcal{A}}^{idRC-b}(1^\lambda)$ est détaillée à la Figure 5.2. Nous avons :

$$\texttt{Adv}_{\mathcal{A}}^{RC} = |\mathbb{P}\big(\texttt{Exp}_{\mathcal{A}}^{RC-1}(1^\lambda) = 1\big) - \mathbb{P}\big(\texttt{Exp}_{\mathcal{A}}^{idRC-0}(1^\lambda) = 1\big)|.$$

Dans le cas de l'expérience $\texttt{Exp}_{\mathcal{A}}^{idRC-b}(1^\lambda)$, l'adversaire $\mathcal{A}'$ ne vote pas à la place de chaque participant corrompu mais énumère leurs choix. Hormis le résultat final du scrutin, il n'apprend aucune autre information. Le dépouillement est ici réalisé par l'algorithme $\texttt{Ideal}-\texttt{Tallying}$ qui agit différemment selon la valeur du bit b. Tout vote émis par un participant honnête est comptabilisé normalement mais, pour ceux générés par l'adversaire, la valeur secrète s_i est étudiée : si $\mathcal{V}_i \notin \mathcal{CV}$ alors le vote n'est pas comptabilisé. Enfin, si b vaut 0 alors l'algorithme ne compte pas les bulletins construits avec s' tandis que si b vaut 1 alors un seul vote associé à s' est comptabilisé.

$\mathtt{Exp}_{\mathcal{A}}^{idRC-b}(1^{\lambda})$

1. $pp \leftarrow \mathtt{Setup}(1^{\lambda})$;
2. $\big(\mathcal{E}(sk_{\mathcal{E}}, pk_{\mathcal{E}}), \mathcal{D}\big((sk_{\mathcal{D}}, pk_{\mathcal{D}}), (sk_{\mathcal{E}}, pk_{\mathcal{E}})\big)\big) \leftarrow \mathtt{KeyGen}(pp)$;
3. $\mathcal{B} \leftarrow \varnothing$, $\mathcal{CV} \leftarrow \mathcal{A}$(liste des votants éligibles) ;
4. Pour tout votant $\mathcal{V}_i \in \mathcal{V}$, $(\sigma_i, s_i) \leftarrow \mathtt{Register}\big(\mathcal{V}_i(pp, ID_i, pk_{\mathcal{E}}), \mathcal{E}(sk_{\mathcal{E}})\big)$;
5. $(j, \beta) \leftarrow \mathcal{A}(\{s_i\}_{i \in \mathcal{CV}})$;
6. Si $\mathcal{CV} \neq n_c$ ou si $j \notin \mathcal{V} \backslash \mathcal{CV}$ ou si $\beta \notin \mathcal{O} \cup \varnothing$ alors retourner 0 ;
7. $b \leftarrow \{0, 1\}$;
8. Si $b = 0$ alors $\mathcal{B} \leftarrow \mathcal{B} \cup B_j$ où $B_j \leftarrow \mathtt{Voting}\big(pp, pk_{\mathcal{D}}, \mathcal{O}, \beta, (\sigma_j, s_j)\big)$;
9. $s'_j \leftarrow s_j$;
10. Pour tout votant $\mathcal{V}_i \in \mathcal{V} \backslash \mathcal{CV}$ avec $i \neq j$,
    $\mathcal{B} \leftarrow \mathcal{B} \cup B_i$ où $B_i \leftarrow \mathtt{Voting}\big(pp, pk_{\mathcal{D}}, \mathcal{O}, \beta, (\sigma_j, s_j)\big)$;
11. $\mathcal{B} \leftarrow \mathcal{B} \cup \tilde{B}_j \cup \{B_i\}_{i \in \mathcal{CV}}$ où $\tilde{B}_j \leftarrow \mathcal{A}(s'_j, \mathcal{B})$ et $B_k \leftarrow \mathtt{Voting}(pp, pk_{\mathcal{D}}, \mathcal{O}, \beta, (\sigma_i, s_i))$
12. $\mathcal{B}' \leftarrow \mathtt{PreVerif}(pp, \mathcal{B}, sk_{\mathcal{E}}, sk_{\mathcal{D}})$
13. $Result \leftarrow \mathtt{Ideal-Tallying}(pp, \mathcal{B}', sk_{\mathcal{D}})$
14. $b' \leftarrow \mathcal{A}(Result)$
15. Retourner $(b' = b)$.

FIGURE 5.2 – Résistance à la coercition, version idéale

5.2 Notre système

Dans cette section, nous détaillons notre propre système de vote électronique résistant à la coercition. Contrairement aux schémas cités auparavant, il permet d'atteindre à la fois une complexité linéaire en nombre de bulletins qui le rend efficace pour de vraies élections, tout en assurant un anonymat maximal et des élections multiples. En outre, toutes les étapes sont vérifiables publiquement. Nous prouvons ensuite la sécurité de notre système et montrons en particulier qu'il satisfait la propriété de résistance à la coercition.

5.2.1 Description

Nous présentons en détail les différentes étapes de notre système de vote électronique : initialisation, génération des clés, enregistrement, vote, pré-vérification et dépouillement comme présentées précédemment.

Initialisation

L'algorithme `Setup` génère les paramètres publics $pp = (p, \mathbb{G}, g_1, g_2, o, O)$ pour une première élection où $\mathbb{G}$ est un groupe cyclique d'ordre premier p, de longueur λ bits et (g_1, g_2, o) sont des générateurs aléatoires de $\mathbb{G}$, où o est choisi pour cette élection en particulier. De même, l'ensemble O est l'ensemble des options de vote possibles, à savoir les candidats pour cette élection. Nous notons $v \in O$, un vote pour un candidat en particulier.

Génération des clés

Les différents participants exécutent `KeyGen` pour construire leurs clés respectives. Les autorités de dépouillement $\mathcal{D}$ partagent une paire de clés $(sk_{\mathcal{D}}, pk_{\mathcal{D}})$ pour le cryptosystème d'ElGamal modifié (voir Section 2.1.2) avec $sk_{\mathcal{D}} = (y_1, y_2)$ où y_1 et y_2 sont aléatoires dans $(\mathbb{Z}_p^*)^2$ et $pk_{\mathcal{D}} = (g_1, g_2, h)$ avec $h = g_1^{y_1} g_2^{y_2}$. Les autorités d'enregistrement $\mathcal{E}$ quant à elles coopèrent pour sélectionner la clé secrète $sk_{\mathcal{E}} = (x_0, x_1)$ aléatoires dans $\mathbb{Z}_p^*$, associée à la valeur publique $pk_{\mathcal{E}} = (C_{x_0}, X_1 = h^{x_1})$, avec $C_{x_0} = g^{x_0} h^x$ pour x aléatoire dans $\mathbb{Z}_p^*$. Cette clé est également partagée par les autorités de dépouillement.

Enregistrement

Après avoir prouvé son éligibilité, le votant exécute le protocole `Register` avec les autorités d'enregistrement et reçoit une accréditation valide unique σ qui dépend d'un secret que lui seul connaît.

Pour cela, les autorités d'enregistrement coopèrent pour choisir deux valeurs aléatoires s dans $\mathbb{Z}_p^*$ et u dans $\mathbb{G}\backslash\{1\}$ puis calculent

$$\sigma = (u, u') \quad \text{où} \quad u' = u^{x_0 + s x_1}.$$

Cette accréditation et le secret s sont transmis au votant par un canal où, par hypothèse, aucune interception n'est possible. Une preuve pour un vérificateur désigné (DVP, voir Section 2.3.2) accompagne ces valeurs et assure que σ a été construite correctement pour le secret s. Par définition, ce type de preuve ne peut convaincre que le destinataire désigné. Par conséquent, elle ne peut pas être utilisée pour attester de la validité de l'accréditation auprès d'un attaquant ou en cas de vente d'accréditation par le votant légitime. En parallèle, σ est également stockée dans une base de données, notée `DB`, qui contient l'ensemble des accréditations valides – sans la valeur s qui est un secret connu uniquement par le votant.

Ainsi, en cas de coercition, un votant peut facilement répondre à un attaquant, sans risque pour son vote réel. Pour cela, il lui suffit de fournir son accréditation σ mais avec un secret s' faux sans que celui-ci ne s'en rende compte. En effet, d'après le Lemme 1 présenté ci-après, un attaquant ne peut pas décider si s' est correct ou non pour σ, sous l'hypothèse DDH. Cette particularité découle des propriétés de notre MAC séquentiellement agrégé : étant donné le triplet (s, u, u'), un attaquant ne peut pas décider si (u, u') est un MAC valide pour s ou non.

Lemme 1. *Etant donné s', $C_{x_0} = g^{x_0} h^x$, $X_1 = h^{x_1}$, $u = h^b$, $u' = u^{x_0 + s x_1}$ avec s, s', x, x_0, x_1, b aléatoires dans $\mathbb{Z}_p^*$ et g, h deux générateurs aléatoires dans $\mathbb{G}$, il est difficile, sous l'hypothèse* DDH, *de décider si $s = s' \mod p$ ou non.*

Démonstration. Nous supposons qu'il existe un oracle qui, à partir de s', $C_{x_0} = g^{x_0} h^x$, $X_1 = h^{x_1}$, $u = h^b$, $u' = u^{x_0 + s x_1}$ avec s, s', x, x_0, x_1, b aléatoires dans $\mathbb{Z}_p^*$ et g, h deux générateurs aléatoires dans $\mathbb{G}$ en entrée, retourne 1 si $s = s' \mod p$ et 0 sinon.

Nous construisons une réduction $\mathcal{R}$ contre l'hypothèse DDH. Par conséquent, nous démontrons comment décider si $c = x_1 b \mod p$ ou non, à partir de $(h, \alpha = h^{x_1}, \beta = h^b, \gamma = h^c)$ pour x_1, b, c aléatoires dans $\mathbb{Z}_p^*$ et g, h deux générateurs aléatoires de $\mathbb{G}$.

La réduction $\mathcal{R}$ fonctionne comme suit. Soit $C_{x_0} = g^{x_0}h^x$ pour x_0, x aléatoires dans $\mathbb{Z}_p^*$. À partir de l'entrée $(h, \alpha, \beta, \gamma)$, elle choisit s' aléatoire dans $\mathbb{Z}_p^*$ puis fournit s', C_{x_0},$X_1 = \alpha$, $u = \beta$ et $u' = u^{x_0}\gamma^{s'}$ à l'oracle. Deux cas sont alors possibles :

1. soit $c = x_1 b \mod p$ et alors $u' = u^{x_0 + s' x_1}$;
2. soit $c \neq x_1 b \mod p$ et donc $c = x_1 b(1 + c')$ pour un $c' \neq 0 \mod p$, d'où $u' = u^{x_0 + s x_1}$ avec $s = s'(1 + c')$, différent de s'.

Par conséquent, l'oracle va déterminer si $s = s'$. Il est alors possible de déduire si $c = x_1 b$ ou non, ce qui casse l'hypothèse DDH.

Dans le cas particulier où s serait égal à 0, même un adversaire disposant d'une puissance de calcul non-bornée ne pourrait pas déterminer si $u = u^{x_0}$ ou non. Cela est due à la propriété d'indistinguabilité parfaite de la mise en gage de Pedersen (voir Définition 33) qui protège de manière inconditionnelle la valeur x_0. □

En outre, la technique générique proposée par ARAÚJO et TRAORÉ [AT13] peut également être utilisée par la suite pour détecter tout vote valide mais néanmoins illégitime car généré à partir d'une accréditation construite par un ensemble d'autorité d'enregistrement qui se seraient associées.

Vote – Première élection

Pour voter, le votant génère un bulletin à partir de l'algorithme `Voting` en utilisant son accréditation σ et son secret. Pour une première élection, le votant calcule une version randomisée σ_r de son accréditation σ, définie comme suit, pour r aléatoire dans $\mathbb{Z}_p^*$:

$$\sigma_r = (u^r, u'^r) = (w, w').$$

Ensuite, le votant choisit son candidat $v \in O$ et génère un bulletin B tel que :

$$B = \langle B_1, B_2, B_3, B_4, B_5, B_6 \rangle = \langle E_{\mathcal{D}}[v], w, w', E_{\mathcal{D}}[w^s], o^s, P \rangle$$

où $E_{\mathcal{D}}[m]$ représente un chiffré M-ElGamal (voir Section 2.1.2) du message m et P est une paire de preuves de connaissance attestant que le bulletin a été correctement construit. Plus précisément, P est constituée des deux preuves suivantes :

- $\pi_1 = PoK\{\alpha : B_4 = E_{\mathcal{D}}[w^\alpha] \wedge B_5 = o^\alpha\}$ prouvant la connaissance du secret s,
- $\pi_2 = PoK\{\beta : B_1 = E_{\mathcal{D}}[\beta] \wedge \beta \in O\}$ prouvant que v appartient bien à l'ensemble O des choix possibles.

Ce bulletin B est envoyé par un canal de communication anonyme à l'urne publique U. En cas de coercition, l'utilisateur utilise un faux secret s' pour émettre le vote selon le choix v' de l'adversaire et génère le bulletin $B' = \langle E_{\mathcal{D}}[v'], w, w', E_{\mathcal{D}}[w^{s'}], o^{s'}, P \rangle$, indistinguable d'un bulletin valide pour quiconque sauf les autorités de dépouillement $\mathcal{D}$.

Pré-vérification

Avant la phase finale de dépouillement, l'algorithme de pré-vérification `PreVerif` permet de vérifier une première fois l'ensemble des bulletins $\mathcal{BB}$ présents dans l'urne. À ce stade, l'utilisation d'accréditations invalides n'est pas considérée : seuls les bulletins dont les calculs

sont invalides ou les doublons sont détectés. Chaque bulletin B dans $\mathcal{BB}$ est vérifié à travers les quatre étapes suivantes, par les autorités de dépouillement $\mathcal{D}$:

Vérification des preuves. Les preuves π_1 et π_2 de la paire P sont vérifiées. Tout bulletin avec au moins une preuve invalide est supprimé.

Suppression des doublons. Les valeurs o^s sont comparées entre elles. Si plusieurs bulletins comportent la même valeur o^s alors cela signifie qu'ils ont été générés avec le même secret s et sont donc des doublons. En accord avec la politique de sécurité définie pour l'élection, les doublons sont supprimés et seul le dernier bulletin soumis est conservé, par exemple.

Reconstruction des accréditations. Les autorités de dépouillement $\mathcal{D}$ coopèrent pour générer le chiffré M-ElGamal $E_{\mathcal{D}}[w]$ de w. À partir des valeurs $E_{\mathcal{D}}[w]$, $E_{\mathcal{D}}[w^s]$ et de la clé secrète $sk_{\mathcal{E}} = (x_0, x_1)$ partagée, elles calculent de manière distribuée les chiffrés $E_{\mathcal{D}}[w^{x_0}]$ et $E_{\mathcal{D}}[(w^s)^{x_1}]$ grâce aux propriétés homomorphes du cryptosystème M-ElGamal. De la même façon, elles coopèrent pour reconstruire une version chiffrée de l'accréditation

$$E_{\mathcal{D}}[w^{x_0+sx_1}] = E_{\mathcal{D}}[w^{x_0}] \cdot E_{\mathcal{D}}[w^{sx_1}].$$

En divisant le troisième membre du chiffré précédent par w', les autorités obtiennent $C = E_{\mathcal{D}}[w^{x_0+sx_1}/w']$. Par conséquent, si l'accréditation (w, w') est valide alors cette valeur C devrait être égale à un chiffré de 1, noté $E_{\mathcal{D}}[1]$.

Pré-test PET. Enfin, un pré-test d'équivalence des clairs (PET, pour *Plaintext Equivalence Test*) [JJ00] est réalisé sur les accréditations. La particularité de ce test est qu'il permet de vérifier si deux chiffrés correspondent au même message clair ou non mais sans le révéler. Pour cela, C est élevé à la puissance α où α est aléatoire dans $\mathbb{Z}_p^*$, calculé conjointement par les autorités de dépouillement $\mathcal{D}$. Ainsi, pour une accréditation σ valide, la valeur $D = C^\alpha$ devrait être égale à $E_{\mathcal{D}}[1^\alpha] = E_{\mathcal{D}}[1]$. A cette étape, D reste néanmoins chiffrée pour éviter toute fuite d'information en particulier en cas de coercition.

Dépouillement

Pour obtenir le résultat final de l'élection, les autorités de dépouillement exécutent l'algorithme `Tallying` qui comporte trois étapes successives :

Mélange. Chaque couple $\langle D, E_{\mathcal{D}}[v]\rangle$ toujours présent suite à la pré-vérification est envoyé à un réseau de mélangeurs, appelé MixNet [Cha81], qui les mélange et les randomise. Le résultat est alors publié sur un registre public U (*Bulletin Board* en anglais), consultable en lecture pour tous. Les bulletins actuels sont donc de la forme

$$B' = \langle D', E'_{\mathcal{D}}[v]\rangle.$$

Identification des votes valides. Pour chaque paire, le chiffré D est déchiffré de manière conjointe par les autorités de dépouillement $\mathcal{D}$. Si le clair obtenu est égal à 1 alors l'accréditation σ_r utilisée et le bulletin associé sont considérés valides. Dans le cas contraire, le bulletin est invalidé et supprimé.

Déchiffrement et comptage des votes. Pour chaque bulletin valide, les autorités de dépouillement coopèrent pour déchiffrer $E'_{\mathcal{D}}[v]$ afin d'obtenir le vote choisi et de procéder au décompte des voix. Le résultat obtenu est alors publié sur l'urne publique U.

Pour toute nouvelle élection ou si des votants ne sont plus éligibles pour celle en cours, les autorités d'enregistrement doivent être capables de mettre à jour les accréditations qui demeurent valides sans obliger les votants éligibles à procéder une nouvelle fois à l'enregistrement. C'est ici qu'intervient notre MAC séquentiellement agrégé pour mettre à jour efficacement ces accréditations.

Elections multiples ou révocation d'accréditations

Les autorités d'enregistrement $\mathcal{E}$ génèrent ensemble un identifiant e_I propre au nouveau scrutin et une nouvelle paire de clés. Ainsi, pour la $i$ème élection, les clés sont définies par $(x_i, X_i = h^{x_i})$ où x_i est aléatoire dans $\mathbb{Z}_p^*$ et partagée par l'ensemble des autorités d'enregistrement et par celles de dépouillement également. Nous détaillons ici le cas d'une seconde élection, identifiée par e_I et pour laquelle la nouvelle paire de clés est $(x_2, X_2 = h^{x_2})$.

Pour chaque accréditation initiale $\sigma = (u, u')$ présente dans la base DB et appartenant à un votant éligible, les autorités d'enregistrement $\mathcal{E}$ sélectionnent conjointement une valeur aléatoire t dans $\mathbb{Z}_p^*$ et calculent alors la nouvelle accréditation σ_2 telle que :

$$\sigma_2 = (u^t, (u' u^{e_I x_2})^t) = (w, w') \quad \text{où} \quad w = w^{x_0 + s x_1 + e_I x_2}.$$

La base de données DB est alors mise à jour avec les nouvelles accréditations et publiée. Chaque votant éligible obtient ainsi sa nouvelle accréditation, associée au même secret s que lui seul connaît. Ces modifications n'influent pas sur les différentes phases du vote électronique. Seule l'étape de pré-vérification nécessite une légère modification pour son troisième critère :

Reconstruction des accréditations. Les autorités de dépouillement $\mathcal{D}$ coopèrent pour générer le chiffré M-ElGamal $E_{\mathcal{D}}[w]$ de w. A partir des valeurs e_I, $E_{\mathcal{D}}[w]$, $E_{\mathcal{D}}[w^s]$ et de la clé secrète $sk_{\mathcal{E}} = (x_0, x_1, x_2)$ partagée, elles calculent les chiffrés $E_{\mathcal{D}}[w^{x_0}]$ et $E_{\mathcal{D}}[(w^s)^{x_1}]$ et $E_{\mathcal{D}}[w^{e_I x_2}]$ grâce aux propriétés homomorphes du cryptosystème M-ElGamal. De la même façon, elles coopèrent pour reconstruire une version chiffrée de l'accréditation

$$E_{\mathcal{D}}[w^{x_0 + s x_1 + e_I x_2}] = E_{\mathcal{D}}[w^{x_0}] \cdot E_{\mathcal{D}}[w^{s x_1}] \cdot E_{\mathcal{D}}[w^{e_I x_2}].$$

En divisant le dernier membre du chiffré précédent par w', les autorités obtiennent $C = E_{\mathcal{D}}[w^{x_0 + s x_1 + e_I x_2} / w']$. Par conséquent, si l'accréditation (w, w') est valide alors cette valeur C devrait être égale à un chiffré de 1, noté $E_{\mathcal{D}}[1]$.

Théorème 9. *Notre système de vote électronique est* publiquement vérifiable*, satisfait la propriété d'*éligibilité *sous l'hypothèse que* $\mathsf{MAC}_{\mathsf{GGM}}$ *est UF-CMVA et est résistant à la coercition sous l'hypothèse* DDH*, dans le modèle de l'oracle aléatoire.*

Démonstration. La preuve du Théorème 9 est disponible ci-après à la Section 5.2.2. □

5.2.2 Preuves de sécurité

Nous prouvons ici le Théorème 9 relatif à la sécurité de notre système de vote électronique, selon le modèle présenté à la Section 5.1.2. De la même façon, nous nous focalisons surtout sur la propriété de résistance à la coercition.

Preuve de la vérifiabilité de bout-en-bout

Pour satisfaire la propriété de vérifiabilité de bout-en-bout, plusieurs conditions doivent être satisfaites :

Vérifiabilité individuelle. À l'enregistrement, le votant $\mathcal{V}$ obtient une preuve DVP pour laquelle il est le vérificateur désigné et qui assure la validité de l'accréditation. Pour voter, $\mathcal{V}$ génère un bulletin qui contient, entre autres, (w, w', o^s) tel que $w = u^r$ et $w' = (u')^r$, où o est un générateur de $\mathbb{G}$ associé à l'élection en cours. Il peut donc chercher ces valeurs dans l'urne U, accessible publiquement. Il s'assure ainsi que son bulletin est présent et sera bien pris en compte puisque toutes les autres phases du vote – pré-vérification et dépouillement – sont également publiquement vérifiables.

Vérifiabilité universelle. Du vote au dépouillement, chaque étape du processus est vérifiable publiquement : le pré-test d'équivalence, le mélange, les preuves de connaissances et la suppression des doublons. Tout le monde peut vérifier que le résultat de l'élection correspond aux bulletins publiés dans l'urne. En particulier, chacun peut vérifier que seuls les bulletins invalides, avec une accréditation ou une preuve invalide, ont été supprimés.

Étant donné que nous supposons que les votes ont été *émis comme prévu* (*cast-as-intended*), notre schéma de vote électronique vérifie bien la propriété de vérifiabilité de bout-en-bout.

Preuve de la propriété d'éligibilité

Comme défini à la Section 5.1.2, nous devons prouver deux conditions distinctes :

Uniquement des votes par des votants éligibles. Cette condition découle directement de la propriété de résistance aux falsifications du schéma $\mathsf{MAC}_{\mathsf{GGM}}$ sous-jacent et du fait que tout bulletin incorrect est supprimé, que ce soit à cause des preuves de connaissance qui ne concordent pas ou d'une accréditation invalide. En outre, nous supposons qu'au moins une autorité d'enregistrement est honnête. Elle refusera donc de collaborer avec les autres pour générer des accréditations valides pour des votants fictifs. Pour détecter de telles accréditations, valides mais illégitimes, il est nécessaire d'utiliser la technique présentée dans [AT13].

Un seul et unique vote par votant. De la même façon, cette condition résulte de la suppression des doublons lors de la phase de pré-vérification. Par conséquent, pour chaque accréditation, il ne peut y avoir qu'un seul vote potentiellement comptabilisé pour le résultat final, selon sa validité.

Preuve de la résistance à la coercition

Soit $\mathcal{A}$ un adversaire contre la propriété de résistance à la coercition de notre système de vote électronique. L'expérience de sécurité correspond à la Figure 5.1 détaillée auparavant. Le but de $\mathcal{A}$ est de déterminer le comportement du votant $\mathcal{V}_j$, cible de la coercition. Soit il a fourni un faux secret et voté pour le candidat désigné par $\mathcal{A}$, soit il a révélé son véritable secret sans émettre de vote. L'adversaire $\mathcal{A}$ gagne avec une probabilité négligeable, comparable à celle de l'adversaire passif $\mathcal{A}'$ dans la version dite idéale, définie à la Figure 5.2.

Nous utilisons la technique de preuves par jeux, formalisée par Shoup [Sho04] en 2004 (voir Section 1.3.2). Nous décrivons en détail le jeu initial et plus brièvement les suivants.

Jeu 0. Le jeu 0 correspond à l'attaque réelle, par l'adversaire $\mathcal{A}$.

1. **Initialisation et génération des clés.** Le challenger $\mathcal{C}$ génère les paramètres publics $pp = (p, \mathbb{G}, g_1, g_2, o, O)$. Il choisit aléatoirement (x_0, x_1) dans $\mathbb{Z}_p^*$ en tant que clé secrète $sk_{\mathcal{E}}$ des autorités d'enregistrement $\mathcal{E}$. La clé publique $pk_{\mathcal{E}} = (C_{x_0}, X_1)$ est publiée, avec $C_{x_0} = g^{x_0}h^x$ et $X_1 = h^{x_1}$ pour x aléatoire dans $\mathbb{Z}_p^*$. De même, $\mathcal{C}$ sélectionne aléatoirement la clé privée $sk_{\mathcal{D}} = (y_1, y_2)$ et calcule $pk_{\mathcal{D}} = (g_1, g_2, h = g_1^{y_1} g_2^{y_2})$.
2. **Enregistrement.** $\mathcal{C}$ simule l'enregistrement effectué par les autorités $\mathcal{E}$. Il génère un ensemble d'accréditations $\sigma_i = (u, u' = u^{x_0 + s_i x_1})$ associées aux secrets s_i pour des votants $\mathcal{V}_i$. $\mathcal{C}$ stocke l'ensemble de ces accréditations et leurs secrets associés. L'ensemble $\mathcal{B}$ des bulletins émis est initialisé comme vide.
3. **Corruption par l'adversaire.** $\mathcal{A}$ sélectionne un ensemble $\mathcal{CV}$ d'électeurs qu'il souhaite corrompre. Il choisit également un votant en particulier $\mathcal{V}_j$ pour la coercition et sélectionne un candidat β. Si l'un de ses choix n'est pas correct, le jeu est abandonné.
4. **Pile ou face.** Le bit b est choisi aléatoirement.
5. **Divulgation des accréditations.** $\mathcal{C}$ fournit à $\mathcal{A}$ l'ensemble des accréditations $\{\sigma_i\}_{i \in \mathcal{CV}}$ des votants corrompus ainsi que $\sigma_j = (u_j, u'_j)$ de la cible sous coercition. Si b vaut 1 alors $s = s_j$ sinon, s est aléatoire et un vote est publié.
6. **Votes honnêtes.** $\mathcal{C}$ génère les bulletins des votants honnêtes et les preuves associées, à partir de la clé publique $pk_{\mathcal{D}} = (g_1, g_2, h)$ et des propriétés homomorphes du chiffrement M-El Gamal. L'ensemble de ces bulletins est noté $\mathcal{B}_{\mathcal{H}}$.
7. **Votes par l'adversaire.** $\mathcal{A}$ émet un ensemble de bulletins pour les votants corrompus. L'ensemble de ces bulletins est noté $\mathcal{B}_{\mathcal{A}}$.
8. **Déchiffrement des votes de l'adversaire.** $\mathcal{C}$ vérifie les preuves associées aux bulletins dans $\mathcal{B}_{\mathcal{A}}$. Nous notons $\mathcal{B}'_{\mathcal{A}}$ l'ensemble de bulletins dont les preuves sont valides. Pour chaque bulletin dans $\mathcal{B}'_{\mathcal{A}}$ et chaque accréditation $\{\sigma_i\}_{i \in \mathcal{CV}} \bigcup \sigma_j$, $\mathcal{C}$ déchiffre les valeurs des bulletins.
9. **Pré-vérification.** $\mathcal{C}$ effectue l'étape de pré-vérification comme suit. La possibilité que $\mathcal{A}$ influe sur un sous-ensemble des autorités est écartée puisqu'une majorité est considérée honnête.

   Vérification des preuves. Soit $\mathcal{B} = \mathcal{B}'_{\mathcal{A}} \cup \mathcal{B}_{\mathcal{H}}$ l'ensemble des bulletins. $\mathcal{C}$ rejette les bulletins dont les preuves de connaissance associées sont invalides. Nous notons $\mathcal{B}_{\pi}$ l'ensemble des bulletins dont les preuves sont valides.

   Suppression des doublons. $\mathcal{C}$ effectue la suppression des doublons.

   Reconstruction des accréditations. À partir des clés (x_0, x_1), $\mathcal{C}$ reconstruit les accréditations en utilisant les propriétés homomorphes d'ElGamal modifié.

   Pré-test. $\mathcal{C}$ réalise le test d'équivalence des accréditations.
10. **Dépouillement.** $\mathcal{C}$ réalise le dépouillement final à l'aide de différents oracles pour le mélange ou encore l'identification des votes valides.

A la fin du jeu, l'adversaire $\mathcal{A}$ retourne un bit b'. Pour S_0 l'évènement "$b' = b$", nous avons alors :

$$\mathbb{P}(S_0) = \mathbb{P}\left(\mathrm{Exp}_{\mathcal{A}}^{RC-b}(1^{\lambda}) = 1\right).$$

Pour chaque jeu i, cet évènement est noté S_i.

Jeu 1 - *Transition basée sur l'hypothèse* DDH. Le jeu 1 est identique au jeu 0, sauf pour la création des bulletins par $\mathcal{C}$.

Le challenger $\mathcal{C}$ choisit une valeur aléatoire τ dans $\mathbb{Z}_p^*$ et calcule $h_1 = g_1^\tau$ et $h_2 = g_2^\tau$. Ainsi, avec (g_1, g_2, h_1, h_2), nous avons un quadruplet Diffie-Helmann. $\mathcal{C}$ crée alors les chiffrés nécessaires aux bulletins comme suit. Pour le choix de candidat $v_k = g_1^{t_k}$, $\mathcal{C}$ choisit un s_i aléatoire dans $\mathbb{Z}_p^*$ et calcule (c_1, c_2, c_3) tels que $c_1 = h_1^{s_i}$, $c_2 = h_2^{s_i}$ et $c_3 = h_1^{s_i y_1} h_2^{s_i y_2} v_k$. Étant donné que (g_1, g_2, h_1, h_2) est un quadruplet Diffie-Hellman, le chiffré respecte la distribution et nous avons alors :

$$\mathbb{P}(S_1) = \mathbb{P}(S_0).$$

Jeu 2 - *Transition basée sur l'extracteur des preuves de connaissance*. Le jeu 2 est identique au jeu 1, sauf pour les bulletins dans $\mathcal{B}_\pi$.

Pour les preuves de connaissance des bulletins dans $\mathcal{B}_\pi$, $\mathcal{C}$ exécute l'extracteur associé aux preuves de connaissance pour extraire le secret s_i de l'accréditation randomisée (w_i, w_i'). Si l'extracteur échoue, $\mathcal{C}$ retourne $\perp$ et stoppe. Nous notons F cet évènement et, par le lemme de différence [Sho04], nous avons :

$$|\mathbb{P}(S_2) - \mathbb{P}(S_1)| \leq \mathbb{P}(F).$$

La probabilité $\mathbb{P}(F)$ est bornée par l'erreur de connaissance des preuves de connaissances qui est négligeable et notée $\nu(k)$. D'où :

$$|\mathbb{P}(S_2) - \mathbb{P}(S_1)| \leq \nu(k).$$

Nous supposons ici que les bulletins sont construits de façon séquentielle et non en parallèle. De cette façon, le challenger $\mathcal{C}$ peut extraire les secrets en rejouant les preuves. Dans le cas parallèle, il faudrait utiliser du chiffrement vérifiable.

Jeu 3 - *Transition basée sur la sécurité de* $\mathsf{MAC}_{\mathsf{GGM}}$. Le jeu 3 est identique au jeu 2, sauf en cas de doute sur un secret et l'accréditation correspondante.

Le challenger $\mathcal{C}$ retourne $\perp$ si au moins un des secrets extraits et son accréditation n'ont pas été fournis par $\mathcal{C}$ ou appartiennent à un votant honnête. Or, nos accréditations sont basées sur $\mathsf{MAC}_{\mathsf{GGM}}$ pour un secret s. Étant donné que le schéma $\mathsf{MAC}_{\mathsf{GGM}}$ est prouvé UF-CMVA, nous avons :

$$|\mathbb{P}(S_3) - \mathbb{P}(S_2)| \leq \mathtt{Adv}^{UF-CMVA}_{\mathcal{A}_{\mathsf{MAC}_{\mathsf{GGM}}}}(1^\lambda).$$

Jeu 4 - *Transition basée sur l'indistinguabilité*. Le jeu 4 est identique au jeu 3, sauf pour le choix du quadruplet (g_1, g_2, h_1, h_2) utilisé pour générer les bulletins des participants honnêtes.

Ici, $\mathcal{C}$ utilise g_1 et g_2 aléatoires pour calculer $h_1 = g^\tau$ et $h_2 = g^\delta$ avec $\tau \neq \delta \bmod p$. Sous l'hypothèse DDH, $\mathcal{A}$ ne peut pas distinguer ce changement. En effet, il est facile de créer un distingueur $\mathcal{D}$ avec un avantage $\mathtt{Adv}^{\mathsf{DDH}}_{\mathcal{D}}(1^\lambda)$ pour le problème DDH. Nous avons alors :

$$|\mathbb{P}(S_4) - \mathbb{P}(S_3)| \leq \mathtt{Adv}^{\mathsf{DDH}}_{\mathcal{D}}(1^\lambda).$$

Aucune information sur les votes n'est divulguée avec ce changement car le cryptosystème M-El Gamal est sémantiquement sûr. En effet, pour tout $\tilde{v} = g_1^{\tilde{t}}$, il existe une clé privée $(\tilde{y}_1, \tilde{y}_2)$

qui correspond à la clé publique h telle que l'algorithme de déchiffrement retourne $\tilde{v}$ pour un chiffré (c_1, c_2, c_3). Nous posons :

- z logarithme discret de h en base g_1 soit $h = g_1^z = g_1^{\tilde{y}_1 g_2^{\tilde{y}_2}}$;
- k logarithme discret de g_2 en base g_1 soit $g_2 = g_1^k$;
- s logarithme discret commun de c_1 et c_2 respectivement en base h_1, h_2 ;
- γ logarithme discret de c_3 en base g_1 soit $c_3 = g_1^\gamma$.

Pour obtenir la clé privée $(\tilde{y}_1, \tilde{y}_2)$, il faut alors résoudre le système d'équations linéaires suivant obtenu en développant les différentes valeurs pour h et c_3 :

$$\begin{cases} \tilde{y}_1 + k\tilde{y}_2 = z \\ \tau s\tilde{y}_1 + \delta s\tilde{y}_2 = \gamma - \tilde{t} \end{cases}$$

Le déterminant de ce système est égal à $\lambda s(\delta - \tau)$ qui est différent de 0 car (g_1, g_2, h_1, h_2) n'est pas un quadruplet DDH c'est-à-dire que $\tau \neq \delta \bmod p$ et, par définition, λ et s sont différents de 0 mod$p$. Ainsi, il existe bien une solution à ce système.

Jeu 5 - *Transition basée sur l'indistinguabilité.* Le jeu 5 est identique au jeu 4, sauf pour le secret envoyé à l'adversaire dans le cas où b vaut 0.

Ici, $\mathcal{C}$ lui envoie un véritable secret s au lieu d'un faux, le vote émis par $\mathcal{A}$ ne sera donc pas comptabilisé. Sous l'hypothèse DDH, $\mathcal{A}$ est ne peut pas faire la différence d'où :

$$|\mathbb{P}(S_5) - \mathbb{P}(S_4)| \leq \mathtt{Adv}_{\mathcal{A}}^{\mathsf{DDH}}(1^\lambda).$$

Au sens de la théorie de l'information, ce jeu ne donne aucune information sur les votes réalisés par des votants honnêtes. La seule information que l'adversaire obtient est le résultat final *Result* des votes honnêtes et le nombre de votes supprimés. Quelque soit la valeur du bit b, $\mathcal{A}$ reçoit le véritable secret et donc une accréditation valide. Nous avons alors :

$$\mathbb{P}(S_5) = \mathbb{P}\left(\mathtt{Exp}_{\mathcal{A}}^{idRC-b}(1^\lambda) = 1\right)$$

Ainsi,

$$\begin{aligned} \mathtt{Adv}_{\mathcal{A}}^{CR}(1^\lambda) &= |\mathbb{P}\left(\mathtt{Exp}_{\mathcal{A}}^{RC-b}(1^\lambda) = 1\right) - \mathbb{P}\left(\mathtt{Exp}_{\mathcal{A}}^{idRC-b}(1^\lambda) = 1\right)| \\ &= |\mathbb{P}(S_0) - \mathbb{P}(S_5)| \end{aligned}$$

Par conséquent, l'avantage de l'adversaire $\mathcal{A}$, pour la propriété de résistance à la coercition, est borné par

$$\begin{aligned} \mathtt{Adv}_{\mathcal{A}}^{CR}(1^\lambda) &\leq \sum_{i=0}^{4} |\mathbb{P}(S_{j+1}) - \mathbb{P}(S_j)| \\ &\leq \nu(k) + \mathtt{Adv}_{\mathcal{A}_{\mathsf{MAC}_{\mathsf{GGM}}}}^{UF-CMVA}(1^\lambda) + 2 \times \mathtt{Adv}^{\mathsf{DDH}}(1^\lambda). \end{aligned}$$

Notre système de vote électronique satisfait la propriété de résistance à la coercition dans le modèle de l'oracle aléatoire, sous les hypothèses DDH et que $\mathsf{MAC}_{\mathsf{GGM}}$ est UF-CMVA.

Conclusion

Dans ce chapitre, nous avons décrit en détail un nouveau schéma de vote électronique à la fois efficace et satisfaisant la propriété de résistance à la coercition. Chaque étape est publiquement vérifiable et l'utilisation de notre MAC séquentiellement agrégé permet de révoquer les accréditations des votants qui ne sont plus éligibles ou de mettre à jour celles des autres votants afin qu'ils puissent l'utiliser lors de nouveaux scrutins. Ce système est prouvé sûr sous l'hypothèse DDH et repose également sur la résistance aux falsifications du schéma $\mathsf{MAC}_{\mathsf{GGM}}$ sous-jacent.

Nous allons maintenant nous intéresser à une deuxième application classique des accréditations anonymes dans le cadre du paiement électronique.

Chapitre 6

Conception d'un système de paiement électronique privé

Dans ce chapitre, nous introduisons un nouveau schéma de post-paiement électronique dit privé, où la banque et le marchand ne sont qu'une seule et même entité. Ce type de paiement couvre différents cas d'usages, comme le télépéage par exemple, et permet d'avoir une garantie forte d'anonymat avec une réelle efficacité. L'utilisateur dispose d'un jeton de paiement qui lui permet d'attester qu'il est enregistré dans le système et de comptabiliser les montants dépensés. Pour cela, nous utilisons notre signature partiellement aveugle, introduite à la Section 3.3. Un seul jeton est suffisant pour une période donnée et pourra être réutilisé un certain nombre de fois, quelque soit le montant, sans risque d'atteinte à l'anonymat de l'utilisateur. Ce schéma est prouvé sûr dans le modèle de l'oracle aléatoire et son implémentation sur carte SIM démontre son efficacité sur des périphériques aux capacités limitées.

Ce résultat a été publié à la conférence FC 2016, dans l'article intitulé *"Private eCash in Practice"* [BBD+17] avec notre signature partiellement aveugle présentée précédemment.

Sommaire

6.1 Paiement électronique

La monnaie électronique a pour but de reproduire les espèces traditionnelles sous forme numérique et offre de nombreux avantages. Un utilisateur peut ainsi dépenser son argent sans les contraintes des billets et des pièces. Il n'est plus nécessaire de disposer de ces éléments physiques, de faire l'appoint ou encore de rendre la monnaie. Trois entités sont représentées : la banque qui émet la monnaie, les utilisateurs qui retirent cette monnaie auprès d'elle et les marchands chez qui les utilisateurs la dépensent.

Dans le plupart des systèmes de paiement électroniques existants, chaque transaction réalisée est connue par la banque. Celle-ci collecte donc une importante quantité de données et peut, à partir de cet historique, extraire de nombreuses informations personnelles. Ainsi, elle peut par exemple apprendre certains des goûts d'un utilisateur, ses loisirs, ses opinions politiques, religieuses ou encore avoir des informations sur d'éventuels problèmes de santé.

En 1982, CHAUM [Cha82] introduit alors la notion de monnaie électronique anonyme où la vie privée de l'utilisateur est garantie vis-à-vis de la banque mais également des marchands. Pour satisfaire cette propriété, il faut pouvoir s'assurer qu'il est impossible de relier entre eux différents évènements que ce soit des retraits ou des dépenses. En outre, le caractère numérique de la monnaie implique de nouveaux risques de sécurité comme la duplication où un même moyen de paiement, appelé jeton, peut alors être utilisé pour plusieurs transactions. Dans le cas non-anonyme, il est facile de comparer l'ensemble des retraits et des dépenses réalisés par un utilisateur. En évitant la traçabilité des opérations, il faut alors garantir la détection des doubles-dépenses ou l'identification des fraudeurs tout en garantissant l'anonymat de l'ensemble des utilisateurs. Pour cela, CHAUM a suggéré l'utilisation des signatures aveugles (voir Section 2.2.1). En outre, pour améliorer l'expérience utilisateur, le système de paiement électronique doit pouvoir être utilisé avec un téléphone mobile ou encore une carte à puce. Ainsi, les protocoles doivent pouvoir être implémentés sur des périphériques aux capacités limitées, en mémoire et en puissance de calcul, tout en respectant un délai de transaction correct.

Dans cette section, nous présentons l'état de l'art principal des systèmes de paiement électronique dit "privé". Nous présentons ensuite la définition formelle ainsi que le modèle de sécurité.

6.1.1 État de l'art

Dans notre cas, nous nous focalisons sur un cas particulier de paiement électronique où il n'existe qu'un marchand géré par une seule entité qui joue également le rôle de la banque. Nous qualifions ce type de paiement électronique de privé. Il caractérise un certain nombre de cas d'usage particulièrement intrusifs comme les transports publics, le télépéage ou encore le rechargement de véhicules électriques. En plus des montants des transactions, ces applications peuvent révéler la localisation de l'utilisateur à un moment donné et conduire à de nouvelles inférences comme le domicile, le lieu de travail ou d'autres activités personnelles. Obtenir un compromis satisfaisant entre les propriétés de sécurité – anonymat, résistance aux falsifications, entre autres – et les performances attendues n'est pas évident. Différentes approches existent pour ces cas d'usages [PBB09, BRT+10, MMCS11] mais nous nous focalisons ici sur les schémas liés au paiement électronique.

Pour le télépéage anonyme, DAY, HUANG, KNAPP et GOLDBERG [DHKG11] introduisent en 2011 deux schémas de paiement respectueux de la vie privée. Le premier n'est que partiellement anonyme puisqu'il nécessite des contrôles qui enregistrent les informations spatio-temporelles de l'utilisateur pour détecter et identifier d'éventuels fraudeurs. En outre, en étudiant l'ensemble des jetons de paiement qui permettent de régler les transactions, il peut être possible de reconstituer les trajets d'un utilisateur. Leur deuxième solution garantit un anonymat total et permet la détection des doubles-dépenses. Cependant, pour être efficace, les

utilisateurs doivent détenir un grand nombre de jetons où chacun ne peut être utilisé qu'à un moment précis. En outre, comme dans le cas des coupons électroniques [LMY15], les tarifs considérés ne sont pas flexibles c'est-à-dire il n'est pas possible de payer des montants différents avec un même jeton.

Dans le cadre du transport public, RUPP, HINTERWÄLDER, BALDIMSTI et PAAR [RHBP13] proposent en 2013 un système où l'utilisateur obtient des jetons de paiement qui valent la valeur maximale qu'il est possible de dépenser dans le système. Un processus dédié pour le remboursement des trop-perçus permet à l'utilisateur de ne payer que pour ses trajets effectifs. En outre, pour assurer l'anonymat des utilisateurs vis-à-vis de la société de transport, dans le modèle honnête mais curieux, des vérifications particulièrement coûteuses sont nécessaires et rendent leur schéma inadapté aux périphériques disposant de capacités de calcul restreintes.

En 2015, MILUTINOVIC, DECROIX, NAESSENS et DE DECKER [MDNDD15] introduisent également un système basé sur des jetons ayant une valeur fixe égale au montant maximum qu'il est possible de dépenser. Néanmoins, les calculs sont également coûteux et moins efficaces car chaque remboursement est fait individuellement sur des jetons distincts.

Cette même année, ARFAOUI, LALANDE, TRAORÉ, DESMOULINS, BERTHOMÉ et GHAROUT [ALT+15] proposent un système de paiement, privé, de titres de transport. Celui-ci respecte le délai attendu d'une transaction – moins de 300 ms – et est efficace, y compris sur des périphériques limités en mémoire et en puissance de calcul. Leur schéma permet également la ré-identification du payeur, nécessaire dans des circonstances particulières et bien définies auparavant. Cependant, ils considèrent que les titres de transports ont tous la même valeur.

En 2014, AU, LIU, FANG, JIANG, SUSILO et ZHOU [ALF+14] s'intéressent au cas du chargement de véhicules électriques. Cependant, leur schéma implique des preuves de connaissance coûteuses. Bien que cela soit acceptable pour le cas du rechargement de véhicules qui nécessite un temps long, cette solution n'est pas applicable lorsque la vitesse de transaction importe, comme précédemment dans le cas du transport.

6.1.2 Définitions

Un système de paiement électronique privé fait intervenir différents participants : un utilisateur $\mathcal{U}$, une société $\mathcal{S}$ et des autorités de révocation $\mathcal{R}$. La société dispose d'un certain nombre de bornes auxquelles elle communique sa clé secrète. Par mesure de sécurité, le rôle d'autorité de révocation est distribué parmi un grand nombre d'entités qui doivent coopérer pour lever l'anonymat d'une transaction.

De façon informelle, les propriétés attendues sont les suivantes :
- chaque utilisateur paye exactement le montant correspondant à ses dépenses ;
- il doit être impossible d'associer une identité particulière à une opération qu'il s'agisse d'un retrait, d'un paiement ou d'une facturation ;
- il doit être impossible d'associer entre eux deux évènements distincts.

Définition 40. Un système de *paiement électronique privé* est constitué des étapes suivantes :

Initialisation. L'initialisation du système est réalisée par l'algorithme `Setup`. Il prend en entrée un paramètre de sécurité λ et retourne les paramètres publics du système.

$$pp \leftarrow \texttt{Setup}(1^\lambda)$$

Génération des clés. L'algorithme de génération des clés, noté KeyGen, est probabiliste. Il prend en entrée les paramètres publics pp et retourne les clés des différents participants : (sk_u, pk_u) pour l'utilisateur, $(sk_{\mathcal{S}}, pk_{\mathcal{S}})$ pour la société et $(sk_{\mathcal{R}}, pk_{\mathcal{R}})$ pour les autorités de révocation. La société $\mathcal{S}$ communique sa paire de clés à l'ensemble des bornes qu'elle possède.

$$\big(\mathcal{U}(sk_u, pk_u), \mathcal{S}(sk_{\mathcal{S}}, pk_{\mathcal{S}}), \mathcal{R}(sk_{\mathcal{R}}, pk_{\mathcal{R}})\big) \leftarrow \texttt{KeyGen}(pp)$$

Enregistrement. Pour s'enregistrer au service, l'utilisateur $\mathcal{U}$ interagit avec la société $\mathcal{S}$ à travers le protocole Register. Il lui fournit son identité ID_u et sa clé publique pk_u. $\mathcal{S}$ enregistre alors le nouvel utilisateur dans une base de données dédiée, notée $\texttt{DB}_{\texttt{REG}}$, et lui fournit un badge $\mathcal{B}_u$. Celui-ci est un support personnel qui réalise les différentes opérations à effectuer par $\mathcal{U}$.

$$\mathcal{U}(\mathcal{B}_u) \leftarrow \texttt{Register}\big(\mathcal{U}(pp, ID_u, pk_u), \mathcal{S}(pp, \texttt{DB}_{\texttt{REG}})\big)$$

Émission de l'accréditation. Pour utiliser le service, l'utilisateur $\mathcal{U}$ doit obtenir une accréditation auprès de la société à travers le protocole TokenIssuance. La société $\mathcal{S}$ fournit à $\mathcal{U}$ un jeton T qui a une valeur maximale M et qui peut être utilisé au plus $N_{\texttt{max}}$ fois. Une transcription $\texttt{transc}_{\texttt{TI}}$ de cet échange est alors stockée dans $\texttt{DB}_{\texttt{REG}}$.

$$\mathcal{U}(T, N_{\texttt{max}}, M) \leftarrow \texttt{TokenIssuance}\big(\mathcal{U}(pp, sk_u, pk_u), \mathcal{S}(pp, (sk_{\mathcal{S}}, pk_{\mathcal{S}}), \texttt{DB}_{\texttt{REG}}, N_{\texttt{max}}, M)\big)$$

Paiement/Contrôle d'accès. À chaque passage auprès d'une borne, l'utilisateur et la borne, notée $\mathcal{S}$ également, s'engagent dans le protocole AccessControl. $\mathcal{U}$ fournit son jeton T à la borne. Elle s'assure que T est valide et permet de s'acquitter de la transaction en cours puis le met à jour avec le montant m correspondant. Si le jeton n'est pas valide, la borne retourne $\perp$ et arrête le protocole. En parallèle, une transcription $\texttt{transc}_{\texttt{AC}}$ de cet échange est sauvegardée dans une base de données dédiée, notée $\texttt{DB}_{\texttt{AC}}$.

$$\mathcal{U}(T, m) \leftarrow \texttt{AccessControl}\big(\mathcal{U}(pp, T), \mathcal{S}(pp, (sk_{\mathcal{S}}, pk_{\mathcal{S}}), \texttt{DB}_{\texttt{AC}}, m))\big)$$

Facturation. Pour payer les transactions effectuées, l'utilisateur $\mathcal{U}$ exécute le protocole TollComputation avec la société $\mathcal{S}$. Il fournit son jeton actuel T et le montant total dépensé M' – il s'agit ici de post-paiement. La société $\mathcal{S}$ vérifie et facture l'utilisateur $\mathcal{U}$ pour le montant M'.

$$M' \leftarrow \texttt{TollComputation}\big(\mathcal{U}(pp, T, M'), \mathcal{S}(pp, (sk_{\mathcal{S}}, pk_{\mathcal{S}}, \texttt{DB}_{\texttt{REG}}))\big)$$

Révocation. Parfois des révocations sont nécessaires, nous en distinguons ici deux types.

Jeton. En cas de vol ou de perte du jeton de paiement, la société $\mathcal{S}$ peut exécuter le protocole $\texttt{Revocation}_T$ avec les autorités de révocation $\mathcal{R}$, suite à une requête de l'utilisateur $\mathcal{U}$ ayant pour identifiant ID_u. À partir d'ID_u et de la transcription $\texttt{transc}_{\texttt{TI}}$, les autorités de révocation $\mathcal{R}$ utilisent leurs clés pour générer les valeurs

V_T nécessaires afin de mettre le jeton associé sur liste noire.

$$V_T \leftarrow \texttt{Revocation}_T\left(\mathcal{S}(ID_u, \texttt{transc}_{\texttt{TI}}), \mathcal{R}(sk_{\mathcal{R}}, pk_{\mathcal{R}})\right)$$

Anonymat. Dans des cas exceptionnels, de sécurité nationale par exemple, il est nécessaire de révoquer l'anonymat assuré aux utilisateurs. Pour cela, la société $\mathcal{S}$ interagit avec les autorités de révocation $\mathcal{R}$ à travers le protocole $\texttt{Revocation}_{ID}$. La société $\mathcal{S}$ fournit la transcription $\texttt{transc}_{\texttt{AC}}$ aux autorités et utilise la base de données d'enregistrement $\texttt{DB}_{\texttt{REG}}$ pour identifier l'utilisateur $\mathcal{U}$ correspondant.

$$ID_u \leftarrow \texttt{Revocation}_{ID}\left(\mathcal{S}(\texttt{transc}_{\texttt{AC}}, \texttt{DB}_{\texttt{REG}}), \mathcal{R}(sk_{\mathcal{R}}, pk_{\mathcal{R}})\right)$$

Modèle de sécurité

Outre la propriété de complétude, un système de paiement électronique privé est dit sûr s'il satisfait les propriétés de résistance aux falsifications, de non-diffamation et de non-chaînabilité. Nous donnons ici leurs définitions à travers des expériences de sécurité entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$.

Nous distinguons deux types d'utilisateurs : ceux qui sont honnêtes, dans l'ensemble $\mathcal{HU}$, dont la clé privée n'est pas divulguée et les utilisateurs corrompus, dans l'ensemble $\mathcal{CU}$, dont $\mathcal{A}$ possède les clés. Cet adversaire $\mathcal{A}$ a accès à certains des oracles suivants :

- $\mathcal{O}_{\texttt{Register}}(ID_u)$ est un oracle utilisé par $\mathcal{A}$ pour enregistrer un nouvel utilisateur honnête. $\mathcal{C}$ exécute la partie utilisateur de l'algorithme `KeyGen`, garde sk_u secrète et envoie pk_u à $\mathcal{A}$. Le couple (ID_u, pk_u) est ajouté à la liste $\mathcal{HU}$.
- $\mathcal{O}_{\texttt{RegisterCorrupt}}(ID_u, (sk_u, pk_u))$ est un oracle utilisé par $\mathcal{A}$ pour enregistrer un utilisateur corrompu avec la paire de clés choisie (sk_u, pk_u). $\mathcal{C}$ n'apprend que la clé publique pk_u et (ID_u, pk_u) est ajouté à la liste $\mathcal{CU}$.
- $\mathcal{O}_{\texttt{Corrupt}}(ID_u)$ est un oracle utilisé par $\mathcal{A}$ pour corrompre un utilisateur honnête, déjà enregistré, dont l'identifiant est ID_u. $\mathcal{C}$ envoie alors sk_u à $\mathcal{A}$. Le couple (ID_u, pk_u) est transféré de $\mathcal{HU}$ vers $\mathcal{CU}$.
- $\mathcal{O}_{\texttt{TokenIssuance}_u}(pk_u)$ est un oracle qui exécute la partie utilisateur du protocole d'émission du jeton `TokenIssuance`. Cet oracle est utilisé par $\mathcal{A}$ jouant le rôle de la société $\mathcal{S}$ avec l'utilisateur $\mathcal{U}$. Si $\mathcal{U}$ accepte de participer au protocole, $\mathcal{A}$ obtient une transcription $\texttt{transc}_{\texttt{TI}}$.
- $\mathcal{O}_{\texttt{TokenIssuance}_{\mathcal{S}}}(pk_u)$ est un oracle qui exécute la partie société du protocole d'émission du jeton `TokenIssuance`. Cet oracle est utilisé par $\mathcal{A}$ jouant le rôle d'un utilisateur corrompu ou non. $\mathcal{A}$ obtient la transcription $\texttt{transc}_{\texttt{TI}}$. Si l'utilisateur simulé est corrompu, $\mathcal{A}$ reçoit également le jeton T généré.
- $\mathcal{O}_{\texttt{AccessControl}_u}(pk_u, m)$ est un oracle qui exécute la partie utilisateur du protocole de contrôle d'accès `AccessControl`. Cet oracle est utilisé par $\mathcal{A}$ jouant le rôle de la société $\mathcal{S}$ avec l'utilisateur $\mathcal{U}$. S'il accepte de participer au protocole, $\mathcal{A}$ obtient la transcription $\texttt{transc}_{\texttt{AC}}$ et le montant m associé. $(\texttt{transc}_{\texttt{AC}}, ID_u)$ est ajouté aux bases de données $\texttt{DB}_{\texttt{REG}}$ et $\texttt{DB}_{\texttt{AC}}$.
- $\mathcal{O}_{\texttt{AccessControl}_{\mathcal{S}}}(pk_u)$ est un oracle qui exécute la partie société du protocole de contrôle

d'accès AccessControl. Cet oracle est utilisé par $\mathcal{A}$ jouant le rôle d'un utilisateur corrompu ou non. $\mathcal{A}$ obtient la transcription $\texttt{transc}_{\texttt{AC}}$ qui est également stockée dans $\texttt{DB}_{\texttt{AC}}$.

- $\mathcal{O}_{\texttt{TollComputation}}(pk_u, T, M')$ est un oracle utilisé par $\mathcal{A}$ pour réaliser l'étape de facturation TollComputation et obtenir le montant final M' dû à $\mathcal{S}$ par l'utilisateur $\mathcal{U}$.
- $\mathcal{O}_{\texttt{Revocation}_T}(pk_u, \texttt{transc}_{\texttt{TI}})$ est un oracle qui exécute la partie autorité de révocation du protocole $\texttt{Revocation}_T$. Cet oracle est utilisé par $\mathcal{A}$ jouant le rôle de la société $\mathcal{S}$. À partir de pk_u et de $\texttt{transc}_{\texttt{TI}}$, $\mathcal{A}$ obtient l'ensemble des passages restants, non utilisés jusqu'à maintenant par l'utilisateur $\mathcal{U}$, dans l'ensemble des $N_{\texttt{max}}$ possibilités.
- $\mathcal{O}_{\texttt{Revocation}_{ID}}(\texttt{transc}_{\texttt{AC}})$ est un oracle qui exécute la partie autorité de révocation du protocole $\texttt{Revocation}_T$. Cet oracle est utilisé par $\mathcal{A}$ jouant le rôle de la société $\mathcal{S}$. À partir de la transcription $\texttt{transc}_{\texttt{AC}}$ obtenue au cours d'un contrôle d'accès, $\mathcal{A}$ obtient l'identité de l'utilisateur $\mathcal{U}$ concerné.

Résistance aux falsifications. La résistance aux falsifications assure qu'il est impossible pour une entité, même une collusion de plusieurs utilisateurs, de (1) réaliser le protocole de paiement AccessControl sans que les autorités de révocation soient capables de retrouver son identité à savoir $\texttt{Revocation}_{ID} = \perp$ ou de (2) réussir à payer moins que ce qu'elle doit réellement.

La propriété est formellement définie par l'expérience de sécurité $\texttt{Exp}^{\text{UF}}_{\mathcal{A}}(1^\lambda)$ décrit dans la Figure 6.1. Le but de l'adversaire est de créer des faux jetons de paiement $T_{(i)}$. Son avantage $\texttt{Adv}^{\text{UF}}_{\mathcal{A}}(1^\lambda)$ est défini par :

$$\texttt{Adv}^{\text{UF}}_{\mathcal{A}}(1^\lambda) = \mathbb{P}\left(\texttt{Exp}^{\text{UF}}_{\mathcal{A}}(1^\lambda) = 1\right).$$

Un système de paiement électronique privé satisfait la propriété de *résistance aux falsifications* si cet avantage est négligeable pour tout adversaire $\mathcal{A}$ polynomial probabiliste.

$\texttt{Exp}^{\text{UF}}_{\mathcal{A}}(1^\lambda)$

1. $pp \leftarrow \texttt{Setup}(1^\lambda)$, $\mathcal{CU} \leftarrow \varnothing$, $\mathcal{HU} \leftarrow \varnothing$;
2. $\left(\mathcal{U}(sk_u, pk_u), \mathcal{S}(sk_{\mathcal{S}}, pk_{\mathcal{S}}), \mathcal{R}(sk_{\mathcal{R}}, pk_{\mathcal{R}})\right) \leftarrow \texttt{KeyGen}(pp)$;
3. $\left(\{T_{(i)}\}_1^n\right) \leftarrow \mathcal{A}^{\mathcal{O}_{\texttt{Register}}, \mathcal{O}_{\texttt{RegisterCorrupt}}, \mathcal{O}_{\texttt{Corrupt}}, \mathcal{O}_{\texttt{TokenIssuance}_{\mathcal{S}}}, \mathcal{O}_{\texttt{AccessControl}_{\mathcal{S}}}, \mathcal{O}_{\texttt{TollComputation}}}()$;
4. Si l'un des $T_{(i)}$ est invalide alors le supprimer ;
5. S'il existe $\texttt{transc}_{\texttt{AC}} \in \texttt{DB}_{\texttt{AC}} : \texttt{Revocation}_{ID}\left(\texttt{transc}_{\texttt{AC}}, \texttt{DB}_{\texttt{AC}}, (sk_{\mathcal{R}}, pk_{\mathcal{R}})\right) = \perp$ alors retourner 1 ;
6. Si $\sum_{i=1}^{n'} \texttt{TollComputation}\left(T_{(i)}, m'_i, (sk_{\mathcal{S}}, pk_{\mathcal{S}}, \texttt{DB}_{\texttt{REG}})\right) < \sum_{i=1}^{|\texttt{DB}_{\texttt{AC}}|} m_i$ alors retourner 1 ;
7. Retourner 0.

FIGURE 6.1 – Résistance aux falsifications

Non-diffamation. La propriété de non-diffamation assure qu'il est impossible pour une entité, comme la société $\mathcal{S}$ même aidée par une collusion d'utilisateurs, de produire une transcription $\texttt{transc}_{\texttt{AC}}$ qui est attribuable à un utilisateur honnête alors qu'il n'en est pas à l'origine.

La propriété est formellement définie par l'expérience de sécurité $\mathtt{Exp}^{\mathrm{ND}}_{\mathcal{A}}(1^\lambda)$ décrit à la Figure 6.2. L'avantage $\mathtt{Adv}^{\mathrm{ND}}_{\mathcal{A}}(1^\lambda)$ de $\mathcal{A}$ est défini par :

$$\mathtt{Adv}^{\mathrm{ND}}_{\mathcal{A}}(1^\lambda) = \mathbb{P}\left(\mathtt{Exp}^{\mathrm{ND}}_{\mathcal{A}}(1^\lambda) = 1\right).$$

$\mathtt{Exp}^{\mathrm{ND}}_{\mathcal{A}}(1^\lambda)$

1. $pp \leftarrow \mathtt{Setup}(1^\lambda)$, $\mathcal{CU} \leftarrow \emptyset$, $\mathcal{HU} \leftarrow \emptyset$;
2. $\left(\mathcal{U}(sk_u, pk_u), \mathcal{S}(sk_{\mathcal{S}}, pk_{\mathcal{S}}), \mathcal{R}(sk_{\mathcal{R}}, pk_{\mathcal{R}})\right) \leftarrow \mathtt{KeyGen}(pp)$;
3. $\mathtt{transc}_{\mathtt{AC}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{Register}}, \mathcal{O}_{\mathtt{RegisterCorrupt}}, \mathcal{O}_{\mathtt{Corrupt}}, \mathcal{O}_{\mathtt{TokenIssuance}_u}, \mathcal{O}_{\mathtt{AccessControl}_u}}(sk_{\mathcal{S}}, \mathtt{DB}_{\mathtt{AC}})$;
4. Si $\mathtt{transc}_{\mathtt{AC}}$ est invalide ou si $\mathtt{Revocation}_{ID}(\mathtt{transc}_{\mathtt{AC}}, \mathtt{DB}_{\mathtt{AC}}, (sk_{\mathcal{R}}, pk_{\mathcal{R}})) = \perp$ alors retourner 0 ;
5. Si $\mathtt{Revocation}_{ID}(\mathtt{transc}_{\mathtt{AC}}, \mathtt{DB}_{\mathtt{AC}}, (sk_{\mathcal{R}}, pk_{\mathcal{R}})) = ID_u \in \mathcal{HU}$ et $(\mathtt{transc}_{\mathtt{AC}}, ID_u) \notin \mathtt{DB}_{\mathtt{AC}}$ alors retourner 1 ;
6. Retourner 0.

FIGURE 6.2 – Non-diffamation

Un système de paiement électronique privé satisfait la propriété de *non-diffamation* si cet avantage est négligeable pour tout adversaire polynomial probabiliste.

Non-chaînabilité. La propriété de *non-chaînabilité* assure qu'il est impossible pour une entité – exceptées les autorités de révocation, même avec l'aide d'utilisateurs malveillants, de relier entre elles : (1) une transcription de l'exécution de `AccessControl` avec une exécution particulière de `TokenIssuance` ou (2) deux transcriptions `AccessControl` réalisées par un même utilisateur ou encore (3) une transcription de l'exécution de `AccessControl` avec une exécution particulière de `TollComputation`.

La propriété est formellement définie par l'expérience de sécurité $\mathtt{Exp}^{\mathrm{NC}-b}_{\mathcal{A}}(1^\lambda)$ décrit à la Figure 6.3. Elle précise que les utilisateurs choisis par l'adversaire $\mathcal{A}$ doivent être enregistrés et honnêtes. De plus, $\mathcal{A}$ peut s'associer avec la société $\mathcal{S}$ et disposer de sa clé secrète $sk_{\mathcal{S}}$. Il peut d'ailleurs faire des requêtes de révocation, d'anonymat ou de jeton, excepté pour les jetons/transcriptions concernés. De même, à l'étape 7, pour la phase de challenge, l'oracle $\mathcal{O}_{\mathtt{TollComputation}}$ est remplacé par $\mathcal{O}_{\mathtt{TollComputation}^*}$ qui ne s'applique que si les conditions suivantes sont vérifiées : (1) les montants m_0 et m_1 associés aux transcriptions sont égaux, (2) l'ensemble des montants utilisés pour des appels à $\mathcal{O}_{\mathtt{AccessControl}}$ sont les mêmes pour u_0 et u_1 et (3) u_0 et u_1 utilisent les mêmes valeurs à leur disposition lors des `AccessControl`.

Ces restrictions peuvent sembler excessives mais reflètent les capacités réelles d'un adversaire. En effet, dans une utilisation concrète avec de nombreux utilisateurs, il est tout à fait probable qu'un certain nombre d'utilisateurs réalisent le même nombre de passages en utilisant des valeurs identiques parmi celles à leur disposition lors des contrôles d'accès. En outre, comme montré par RUPP, BALDIMSTI, HINTERWÄLDER et PAAR [RBHP15], le montant agrégé ne doit pas permettre de retrouver les différents montants dépensés par l'utilisateur.

L'avantage $\mathtt{Adv}^{\mathrm{NC}-b}_{\mathcal{A}}$ de $\mathcal{A}$ est défini par :

$$\mathtt{Adv}^{\mathrm{NC}-b}_{\mathcal{A}} = |\mathbb{P}\left(\mathtt{Exp}^{\mathrm{NC}-1}_{\mathcal{A}}\right) - \mathbb{P}\left(\mathtt{Exp}^{\mathrm{NC}-0}_{\mathcal{A}}\right)|.$$

Un système de paiement électronique privé satisfait la propriété de *non-chaînabilité* si cet avantage est négligeable pour tout adversaire $\mathcal{A}$ polynomial probabiliste.

$\mathrm{Exp}_{\mathcal{A}}^{\mathrm{NC}-b}(1^{\lambda})$

1. $pp \leftarrow \mathtt{Setup}(1^{\lambda})$, $\mathcal{CU} \leftarrow \varnothing$, $\mathcal{HU} \leftarrow \varnothing$;
2. $\big(\mathcal{U}(sk_u, pk_u), \mathcal{S}(sk_{\mathcal{S}}, pk_{\mathcal{S}}), \mathcal{R}(sk_{\mathcal{R}}, pk_{\mathcal{R}})\big) \leftarrow \mathtt{KeyGen}(pp)$;
3. $(u_0, u_1, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{Register}}, \mathcal{O}_{\mathtt{RegisterCorrupt}}, \mathcal{O}_{\mathtt{Corrupt}}, \mathcal{O}_{\mathtt{TokenIssuance}_u}, \mathcal{O}_{\mathtt{AccessControl}_u}, \mathcal{O}_{\mathtt{Revocation}_{ID}}, \mathcal{O}_{\mathtt{Revocation}_T}, \mathcal{O}_{\mathtt{TollComputation}}}(sk_{\mathcal{S}}, sk_{\mathcal{R}}, \mathtt{DB}_{\mathtt{AC}}, \mathtt{DB}_{\mathtt{AC}}, \mathtt{DB}_{\mathtt{REG}})$;
4. Si u_o ou u_1 corrompu alors retourner 0 ;
5. Pour $i \in \{0,1\}$,
   $(T_{(i)}, \mathtt{transc}_{\mathtt{AC}}^{(i)}) \leftarrow \mathtt{TokenIssuance}\big(\mathcal{C}(pp, sk_{u_i}, pk_{u_i}), \mathcal{S}(pp, sk_{\mathcal{S}}, pk_{\mathcal{S}}, \mathtt{DB}_{\mathtt{REG}}, N_{\max}, M)\big)$
6. $\mathtt{transc}_{\mathtt{AC}}^{(b)} \leftarrow \mathtt{AccessControl}\big(\mathcal{C}(pp, T_{(1)}), \mathcal{S}(pp, (sk_{\mathcal{S}}, pk_{\mathcal{S}}), \mathtt{DB}_{\mathtt{AC}}, m_b)\big)$
   et $\mathtt{transc}_{\mathtt{AC}}^{(1-b)} \leftarrow \mathtt{AccessControl}\big(\mathcal{C}(pp, T_{(0)}), \mathcal{S}(pp, (sk_{\mathcal{S}}, pk_{\mathcal{S}}), \mathtt{DB}_{\mathtt{AC}}, m_{1-b})\big)$
7. $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{Register}}, \mathcal{O}_{\mathtt{RegisterCorrupt}}, \mathcal{O}_{\mathtt{Corrupt}}, \mathcal{O}_{\mathtt{TokenIssuance}_u}, \mathcal{O}_{\mathtt{AccessControl}_u}, \mathcal{O}_{\mathtt{Revocation}_{ID}}, \mathcal{O}_{\mathtt{Revocation}_T}, \mathcal{O}_{\mathtt{TollComputation}^*}}(sk_{\mathcal{S}}, sk_{\mathcal{R}}, \mathtt{DB}_{\mathtt{AC}}, \mathtt{DB}_{\mathtt{AC}}, \mathtt{DB}_{\mathtt{REG}})$;
8. Si $\mathcal{O}_{\mathtt{Corrupt}}(ID_{u_0})$ ou $\mathcal{O}_{\mathtt{Corrupt}}(ID_{u_1})$ alors retourner 0 ;
9. Si $\mathcal{O}_{\mathtt{Revocation}_{ID}}(\mathtt{transc}_{\mathtt{AC}}^{(b)})$ ou $\mathcal{O}_{\mathtt{Revocation}_{ID}}(\mathtt{transc}_{\mathtt{AC}}^{(1-b)})$ alors retourner 0 ;
10. Si $\mathcal{O}_{\mathtt{Revocation}_T}(pk_{u_0}, \mathtt{transc}_{\mathtt{AC}}^{(0)})$ ou $\mathcal{O}_{\mathtt{Revocation}_T}(pk_{u_1}, \mathtt{transc}_{\mathtt{AC}}^{(1)})$ alors retourner 0 ;
11. Retourner $(b' = b)$.

FIGURE 6.3 – Non-Chaînabilité

À noter qu'ici, la propriété de non-chaînabilité entraîne celle d'anonymat. Cette dernière assure qu'aucune entité – exceptée les autorités de révocation, même aidée par des utilisateurs malveillants, ne doit pouvoir identifier un utilisateur derrière une transcription donnée.

6.2 Notre système

Dans cette section, nous détaillons notre propre système de paiement électronique privé entre un utilisateur $\mathcal{U}$, la société $\mathcal{S}$ et les autorités de révocation $\mathcal{R}$.

Remarques : 1. Aucune entité n'est supposée de confiance puisqu'elles peuvent toutes avoir intérêt à tricher. Seul le badge fourni à l'utilisateur au cours de l'enregistrement est supposé inviolable – *tamper-resistant* : tous les calculs qu'ils réalisent sont corrects et toute tentative pour l'altérer est détectée.
2. Pour être utilisable pour différents cas d'usage, le système doit respecter une contrainte de délai particulière. Ainsi, l'objectif est de réaliser des transactions en moins de 300ms [ALT+15]. Pour cela, il faut prendre en compte les capacités limitées des périphériques comme la carte SIM qui ne peut pas réaliser d'opérations complexes ou coûteuses comme les calculs de couplage.

6.2.1 Description

Nous présentons en détail les différentes étapes de notre schéma de paiement électronique privé : initialisation, génération des clés, enregistrement, émission de l'accréditation, contrôle

d'accès, facturation et révocations.

Dans le cas du télépéage par exemple, $\mathcal{S}$ serait une société d'autoroutes qui possède un certain nombre de bornes de péage sur son réseau. Un utilisateur enregistré au service de télépéage doit donc montrer à chaque péage qu'il possède un badge valide lui permettant de payer pour ce trajet. À chaque passage validé, son jeton présent dans le badge est mis à jour et additionne les montants particuliers des péages passés. De cette façon, les trajets effectués sont inconnus de la société d'autoroutes et l'utilisateur est facturé pour l'ensemble des trajets qu'il aura effectués, à la fin de la période choisie.

Initialisation

L'algorithme `Setup` génère les paramètres publics $pp = (q, \mathbb{G}, g, h, g_R, N_{\texttt{max}}, \{g_i\}_{i=0}^{N_{\texttt{max}}})$ où $\mathbb{G}$ est un groupe cyclique d'ordre premier q, de longueur λ bits, et $(g, h, g_R, \{g_i\}_{i=0}^{N_{\texttt{max}}})$ sont des générateurs aléatoires de $\mathbb{G}$ avec $N_{\texttt{max}}$ qui désigne le nombre maximum d'utilisations du jeton de paiement, en accord avec les besoins de l'utilisateur.

Génération des clés

L'algorithme `KeyGen` fournit les clés aux différents participants. Chaque utilisateur potentiel $\mathcal{U}$ obtient une paire de clés (sk_u, pk_u) qui permet de l'authentifier.

La société $\mathcal{S}$ détient la clé secrète $\vec{sk} = (x_0, x_1, x_2)$ qu'elle communique à chaque borne. Par conséquent, nous noterons indifféremment les bornes et la société par $\mathcal{S}$. Les paramètres publics associés sont $C_{x_0} = g^{x_0} h^{\tilde{x}_0}$ qui est une mise en gage de x_0 aléatoire dans $\mathbb{Z}_q^*$, $X_1 = h^{x_1}$ et $X_2 = h^{x_2}$. Elles détiennent aussi une paire de clés $(sk_{\mathcal{S}}, pk_{\mathcal{S}})$ du schéma de signature RSA, utilisée pour certains échanges.

Les autorités de révocation $\mathcal{R}$ coopèrent pour générer deux paires de clés : $(sk_{\mathcal{R}_E}, pk_{\mathcal{R}_E})$ pour le cryptosystème ElGamal à seuil et $(sk_{\mathcal{R}_P}, pk_{\mathcal{R}_P})$ pour le cryptosystème de Paillier à seuil (voir Section 2.1.2). Les clés privées sont donc partagées parmi les autorités de révocation [GJKR03, FS01]. Au moins t d'entre elles doivent coopérer pour identifier un utilisateur ou révoquer un jeton.

Pour atteindre la propriété de résistance aux falsifications, y compris dans un environnement où l'adversaire peut interagir avec la société $\mathcal{S}$, le chiffrement de Paillier est utilisé comme mise en gage extractible [HT07].

Soit g_E un générateur de $\mathbb{G}$. La clé privée ElGamal $sk_{\mathcal{R}_E}$ est x_T, aléatoire dans $\mathbb{Z}_p^*$ et la clé publique associée $pk_{\mathcal{R}_E}$ est égale à (g_E, X_T) où $X_T = g_E^{x_T}$. Soit g_P un générateur de $\mathbb{G}$. La clé privée de Paillier $sk_{\mathcal{R}_P}$ est (a, b) où a et b sont deux nombres premiers distincts tirés aléatoirement tels que $|a| = |b|$ et $\text{pgcd}(ab, (a-1)(b-1)) = 1$. La clé publique $pk_{\mathcal{R}_P}$ associée est (g_P, n) avec $n = ab$.

Enregistrement

Pour utiliser le service, $\mathcal{U}$ exécute le protocole `Register` avec la société $\mathcal{S}$. Il lui fournit sa clé publique pk_u et une preuve à divulgation nulle de connaissance attestant qu'il connaît bien la clé secrète sk_u associée. Si la preuve est valide, $\mathcal{U}$ obtient un badge personnel $\mathcal{B}_u$ qui contient

une carte SIM. Celui-ci permet à $\mathcal{U}$ d'utiliser le service de façon anonyme. En parallèle, pk_u est enregistré dans une base de données dédiée, notée $\mathtt{DB}_{\mathtt{REG}}$.

Émission de l'accréditation

À chaque nouvelle période de facturation, l'utilisateur enregistré $\mathcal{U}$ initialise le protocole `TokenIssuance` auprès de la société $\mathcal{S}$ à travers une requête signée avec la signature de Schnorr (voir Section 2.2.1). La Figure 6.4 présente l'ensemble des échanges, détaillés ci-après.

Utilisateur $\mathcal{U}(pp, sk_u, C_{x_0}, X_1, X_2)$		Société $\mathcal{S}(pp, pk_u, (x_0, x_1, x_2), C_{x_0}, X_1, X_2, \mathtt{DB}_{\mathtt{REG}})$
$r, r_1, s \xleftarrow{R} \mathbb{Z}_q^*$ $C_s \leftarrow h^r X_1^s$ $W \leftarrow g^s$ $D \leftarrow g_P^s r_1^n$ Construire π_1	$\xrightarrow{C_s, W, D, \pi_1}$	Vérifier π_1 $s', b \xleftarrow{R} \mathbb{Z}_q^*$ $u \leftarrow h^b$, $u'' \leftarrow u^{x_0}(C_s X_1^{s'})^b$
Vérifier π_2	$\xleftarrow{g^{s'}, (u, u''), \pi_2}$	Construire π_2
$\sigma \leftarrow Sign_{sk_u}(W, D, g^{s'})$	$\xrightarrow{\sigma, (W, D, g^{s'})}$	Vérifier σ
$u' \leftarrow \frac{u''}{u^r}$, $m \leftarrow 0$	$\xleftarrow{s'}$	Stocker (W, D, s', σ) dans $\mathtt{DB}_{\mathtt{REG}}$
$T \leftarrow (u, u')$, $F \leftarrow \{g_i\}_{i=0}^{N_{\max}}$		
$s_u \leftarrow s + s'$		

FIGURE 6.4 – Protocole `TokenIssuance` de notre système de paiement électronique privé

Tout d'abord, $\mathcal{U}$ choisit une valeur aléatoire s secrète qu'il met en gage dans C_s et calcule W tels que

$$C_s = h^r X_1^s, W = g^s.$$

Il calcule également un chiffré de Paillier $D = g_P^s r_1^s$ qui sera utilisé par les autorités uniquement en cas de nécessité de révocation. L'utilisateur $\mathcal{U}$ envoie (C_s, W, D) accompagné d'une preuve de connaissance π_1 qui assure que ces valeurs ont été calculées correctement, définie comme suit :

$$\pi_1 = \mathrm{PoK}\{\alpha, \beta, \gamma : C_s = h^\alpha X_1^\beta \wedge W = g^\beta \wedge D = g_P^\beta \gamma^n\}.$$

Après vérification et à partir de sa clé privée $\vec{sk}$, la société $\mathcal{S}$ fournit à $\mathcal{U}$ une paire (u, u''), pour b et s' aléatoires dans $\mathbb{Z}_q^*$, définie par :

$$\left(u = h^b, u'' = u^{x_0}(C_s X_1^{s'})^b\right).$$

$\mathcal{S}$ fournit également une preuve de connaissance π_2 qui assure que les calculs sont corrects :

$$\pi_2 = \mathrm{PoK}\{\alpha, \beta, \gamma : u = h^\alpha \wedge u'' = u^\beta (C_s X_1^{s'})^\alpha \wedge C_{x_0} = h^\beta h^\gamma\}.$$

En développant, nous avons u'' égal à $u^r(u^{x_0+x_1(s+s')})$. Pour $s_u = s+s'$, $\mathcal{U}$ obtient donc une signature partiellement aveugle sur le message s_u. Par la suite, une information commune m sera ajoutée pour stocker le montant dépensé avec le jeton.

Finalement, l'utilisateur génère une signature de ces valeurs $(W, D, g^{s'})$ que $\mathcal{S}$ stocke dans une base de données dédiée $\mathtt{DB}_{\mathtt{REG}}$. Elle sera utilisée en cas de révocation et pour le calcul du montant à la fin de la période. En divisant par u^r, $\mathcal{U}$ obtient son jeton de paiement T qui est :

$$T = \left(u = h^b, u' = u^{x_0+x_1 s_u+x_2 m}\right).$$

Ce jeton a une valeur maximale choisie notée M et peut être utilisé au maximum $N_{\mathtt{max}}$ fois. Ainsi, un ensemble F est initialisé à $\{g_i\}_{i=0}^{N_{\mathtt{max}}}$. A chaque usage du jeton, un g_i distinct sera utilisé et servira à déterminer le nombre d'utilisations. Cet ensemble F est mis à jour à chaque nouvelle période de facturation afin d'éviter tout risque de traçage d'un utilisateur.

Contrôle d'accès

À chaque borne, l'utilisateur $\mathcal{U}$ s'engage dans le protocole `AccessControl`. Pour rappel, cette borne est notée $\mathcal{S}$ puisqu'elle appartient à la société et connaît ses clés. La Figure 6.5 détaille l'ensemble des calculs présentés ci-après.

Pour être autorisé à passer, $\mathcal{U}$ fournit un version randomisée (w, w') de son jeton T, un chiffré ElGamal de g^{s_u} ainsi que différentes valeurs c', c_1, c_2, V, A et T_i nécessaires à $\mathcal{S}$ pour vérifier la validité du jeton ou le mettre à jour. $\mathcal{U}$ envoie ces valeurs accompagnées de la preuve π_3 qui atteste qu'elles sont correctes :

$$\begin{aligned}\pi_3 \quad = \quad & \mathrm{PoK}\{\alpha,\beta,\gamma,\delta,\sigma,\mu,\eta : c_1 = w^\sigma h^\beta \wedge c_2 = w^\mu h^\gamma \wedge V = g^{-\alpha} X_1^\beta X_2^\gamma \\ & \wedge A = h^\delta X_1^\sigma X_2^\mu \wedge T_i = g_i^\sigma \wedge E_1 = g_E^\eta \wedge E_2 = g^\sigma X_T^\eta\}.\end{aligned}$$

Après réception, la borne $\mathcal{S}$ vérifie la validité du jeton. À partir de la clé secrète $\vec{sk} = (x_0, x_1, x_2)$, elle vérifie la valeur V avec (c_1, c_2, c') puis la preuve π_3. Si ces vérifications sont correctes, elle met à jour le jeton avec le montant m' correspondant au coût de la transaction en cours. Pour cela, la borne calcule la paire (y, y''), pour d aléatoire dans $\mathbb{Z}_q^*$:

$$\left(y = h^d, y'' = y^{x_0}(AX_2^{m'})^d\right).$$

En développant, y'' est égal à $y^t(y^{x_0+x_1 s_u+x_2(m+m')})$. Nous notons que le montant m, connu uniquement par l'utilisateur, a été agrégé avec m' qui correspond au montant de la transaction passée. Ainsi, la mise à jour est réalisée avec notre signature partiellement aveugle sur le secret s_u et l'information commune m'. $\mathcal{S}$ stocke alors (E, T_i) dans la base de données $\mathtt{DB}_{\mathtt{AC}}$ pour les vérifications futures ou une éventuelle révocation puis envoie y, y'' et m' à $\mathcal{U}$ avec une preuve de connaissance π_4 définie comme suit :

$$\pi_4 = \mathrm{PoK}\{\alpha,\beta,\gamma : y = h^\alpha \wedge y'' = y^\beta (AX_2^{m'})^\alpha \wedge C_{x_0} = g^\beta h^\gamma\}.$$

À noter que la valeur T_i dans $\mathtt{DB}_{\mathtt{AC}}$ peut être utilisée a posteriori pour vérifier une éventuelle fraude. En effet, à chaque passage, l'utilisateur utilise un g_i distinct dans F pour $T_i = g_i^{s_u}$. Cette valeur permet alors de s'assurer que le jeton T n'a pas utilisé plus de $N_{\mathtt{max}}$ fois ou deux fois

Utilisateur $\mathcal{U}(pp, X_1, X_2, pk_{\mathcal{S}}, (u, u'), s_u, m, F)$		Borne $\mathcal{S}(pp, sk_{\mathcal{S}}, (x_0, x_1, x_2), m', \mathtt{DB_{AC}})$
$l, r, z_1, z_2, t, a \overset{R}{\leftarrow} \mathbb{Z}_q^*$, $g_i \overset{R}{\leftarrow} F$ $w \leftarrow u^l$, $w' \leftarrow (u'^l)$, $c' \leftarrow w' g^r$ $c_1 \leftarrow w^{s_u} h^{z_1}$, $c_2 \leftarrow w^m h^{z_2}$ $V \leftarrow g^{-r} X_1^{z_1} X_2^{z_2}$, $T_i \leftarrow g_i^{s_u}$ $A \leftarrow h^t X_1^{s_u} X_2^m$, $F \leftarrow F \backslash \{g_i\}$ $E \leftarrow (E_1 \leftarrow g_E^a, E_2 \leftarrow g^{s_u} X_T^a)$		
Construire π_3	$\xrightarrow{w, c', c_1, c_2, V}$ $\xleftarrow{T_i, A, g_i, \pi_3, E}$	Tester si $V = \frac{w^{x_0} c_1^{x_1} c_2^{x_2}}{c'}$ Vérifier π_3 $d \overset{R}{\leftarrow} \mathbb{Z}_q^*$ $y \leftarrow h^d$, $y'' \leftarrow y^{x_0} (A X_2^{m'})^d$ Construire π_4
Vérifier S (et π_4) $y' \leftarrow \frac{y''}{y^t}$ $m \leftarrow m + m', T \leftarrow (y, y')$	$\xleftarrow{(m', y, y'', \pi_4), S}$	$S \leftarrow Sign_{RSA}(m', y, y'', \pi_4)$ Stocker (E, T_i) dans $\mathtt{DB_{AC}}$

FIGURE 6.5 – Protocole `AccessControl` de notre système de paiement électronique privé

avec le même g_i. Les badges considérés sont ici supposés inviolables et n'utilisent pas deux fois le même g_i. Dans le cas où un badge est compromis, la valeur E sera utilisée pour l'identifier comme nous le verrons par la suite.

En outre, pour respecter la contrainte de délai imposée par le passage à la borne, la preuve π_4 ne peut pas être vérifiée par l'utilisateur dès réception. Par conséquent, la borne fournit une signature RSA, notée S, des valeurs envoyées (y, y'', m', π_4). $\mathcal{U}$ peut vérifier rapidement la signature S pour un exposant RSA de vérification choisi petit. La preuve π_4 sera quant-à-elle vérifiée par la suite par la carte SIM lors d'un temps d'inactivité. Enfin, en divisant par y^t, $\mathcal{U}$ obtient son nouveau jeton T qui devient :

$$T = \left(y = h^d, y' = y^{x_0 + x_1 s_u + x_2 (m + m')}\right).$$

Pour atteindre une non-chaînabilité totale, il est possible d'ajouter de l'aléa à la valeur T_i à partir d'une fonction pseudo-aléatoire. Pour plus de détails, nous invitons le lecteur à se référer à [DY05].

Facturation

À la fin de la période de facturation, l'utilisateur exécute le protocole `TollComputation` auprès de la société $\mathcal{S}$ pour payer l'ensemble des passages effectués aux bornes.

Dans le cas où il n'a pas utilisé son jeton $N_{\mathtt{max}}$ fois, il doit tout d'abord simuler les utilisations restantes à travers le protocole présenté à la Figure 6.6. Il s'agit d'une émulation de l'algorithme `AccessControl`, où le montant m' sera égal au montant maximal M du jeton. Plus précisément, tout comme pour le protocole `AccessControl`, $\mathcal{U}$ calcule une version randomisée (w, w') de son jeton, les valeurs (c', c_1, c_2, V, A) et, pour chaque g_i restant dans F, il

Utilisateur $\mathcal{U}(pp, X_1, X_2, pk_{\mathcal{S}}, (u, u'), s_u, m, F)$		Borne $\mathcal{S}(pp, sk_{\mathcal{S}}, (x_0, x_1, x_2), M, \mathtt{DB}_{\mathtt{AC}})$
$l, r, z_1, z_2, t, a \xleftarrow{R} \mathbb{Z}_q^*$, $g_i \xleftarrow{R} F$		
$w \leftarrow u^l$, $w' \leftarrow (u'^l)$, $c' \leftarrow w' g^r$		
$c_1 \leftarrow w^{s_u} h^{z_1}$, $c_2 \leftarrow w^m h^{z_2}$		
$V \leftarrow g^{-r} X_1^{z_1} X_2^{z_2}$		
$A \leftarrow h^t X_1^{s_u} X_2^m$, $F \leftarrow F \backslash \{g_i\}$		
Pour tout g_i dans F, $T_i \leftarrow g_i^{s_u}$		
Construire π'_3	$\xrightarrow{w, c', c_1, c_2, V, \{g_i\}_F}$	Tester si $V = \frac{w^{x_0} c_1^{x_1} c_2^{x_2}}{c'}$
	$\xrightarrow{\{T_i\}_{g_i \in F}, A, \pi'_3, E}$	Vérifier π'_3
		$d \xleftarrow{R} \mathbb{Z}_q^*$
		$y \leftarrow h^d$, $y'' \leftarrow y^{x_0} (A X_2^{\lvert F \rvert M})^d$
		Construire π'_4
Vérifier S (et π'_4)	$\xleftarrow{(\lvert F \rvert m', y, y'', \pi'_4), S}$	$S \leftarrow Sign_{RSA}(\lvert F \rvert M, y, y'', \pi_4)$
$y' \leftarrow \frac{y''}{y^t}$		Stocker (E, T_i) dans $\mathtt{DB}_{\mathtt{AC}}$
$m \leftarrow m + \lvert F \rvert M$, $T \leftarrow (y, y')$		

FIGURE 6.6 – Emulation d'`AccessControl` pour utiliser tous les passages autorisés

génère $T_i = g_i^{s_u}$. La preuve π'_3 est identique à la preuve π_3 mais elle prend en compte plusieurs T_i à la fois. Après les différentes vérifications, $\mathcal{S}$ génère (y, y'') où elle agrège autant de fois la valeur M que de T_i reçus, soit $|F|$, et les stocke également dans sa base de données $\mathtt{DB}_{\mathtt{AC}}$. Ainsi, après vérification, $\mathcal{U}$ obtient comme jeton final :

$$T = \left(y = u^d, y' = y^{x_0 + x_1 s_u + x_2 (m + |F|M)}\right).$$

Ces échanges ont permis à $\mathcal{U}$ de réaliser au total $N_{\mathtt{max}}$ dépenses avec son jeton mais en ne comptant malgré tout que les passages vraiment effectués, grâce à notre signature partiellement aveugle.

$\mathcal{U}$ s'engage alors dans le protocole `TollComputation` avec la société pour être facturé selon l'utilisation de son jeton au cours de la période. Pour cela, il génère une version randomisée (R, R') de son jeton et les valeurs (c', c_1, V, T_g) nécessaires pour les vérifications avec $T_g = g_R^{s_u}$, comme lors d'un contrôle d'accès. La valeur c_2 n'est pas utile puisque le montant m n'est pas mis en gage, tout comme la valeur A puisque le jeton ne sera pas mis à jour. $\mathcal{U}$ envoie donc (m, R, c', c_1, V, T_g) avec la preuve π_5 définie comme suit :

$$\pi_5 = \text{PoK}\{\alpha, \beta, \gamma : T_g = g_R^{\alpha} \wedge c_1 = R^{\alpha} h^{\beta} \wedge V = g^{-\gamma} X_1^{\beta}\}.$$

Après réception, $\mathcal{S}$ s'assure que V est correct avec sa clé secrète $\vec{sk}$ et vérifie π_5. Il vérifie également que T_g n'est pas déjà présent dans la base de données $\mathtt{DB}_{\mathtt{REG}}$. Cela permet de s'assurer que l'utilisateur n'a pas déjà réalisé cette opération. Le montant m correspond au montant total à payer.

Si l'utilisateur ne réalise pas le protocole de facturation, en accord avec la politique du

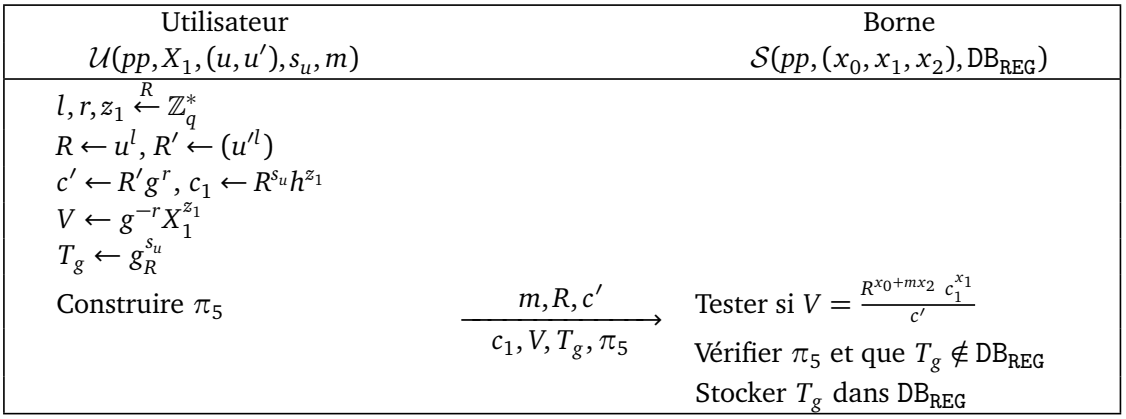


FIGURE 6.7 – Protocole `TollComputation` de notre système de paiement électronique privé

service, il peut être amené à devoir payer le montant maximum qu'il est possible de régler avec un jeton au cours d'une période, à savoir $N_{\mathtt{max}} \times M$.

Révocation

En cas de nécessité, deux types de révocations sont possibles.

Révocation d'un jeton. En cas de perte ou du vol du badge qui sert de support au jeton, il est possible de le révoquer. Pour cela, la société $\mathcal{S}$ fournit le chiffré de Paillier D du secret s aux autorités de révocation, obtenu lors de la phase d'obtention du jeton. Elles collaborent alors pour déchiffrer D et obtiennent s. Ainsi, $\mathcal{S}$ peut calculer $s_u = s + s'$ et mettre toutes les valeurs $T_i = g_i^{s_u}$ pour i allant de 1 à $N_{\mathtt{max}}$ sur liste noire.

Révocation d'anonymat. Le but est ici d'identifier l'auteur d'un accès donné, pour des raisons exceptionnelles de sécurité nationale, par exemple. Dans ce cas, la société $\mathcal{S}$ fournit le chiffré ElGamal E de g^{s_u} obtenu au cours du passage ciblé aux autorités de révocation $\mathcal{R}$. Au moins t d'entre elles doivent coopérer pour retrouver la valeur g^{s_u}. En utilisant les données stockées lors de la phase d'obtention du jeton, il est alors facile de retrouver l'utilisateur correspondant.

Théorème 10. *Notre schéma de paiement électronique privé est* résistant aux falsifications *sous l'hypothèse que* $\mathsf{MAC}_{\mathsf{GGM}}$ *est UF-CMVA, satisfait la propriété de* non-diffamation *sous l'hypothèse* DL *et est* non-chaînable *sous les hypothèses* DCR, DDH *et* $q-$DDHI, *dans le modèle de l'oracle aléatoire.*

Démonstration. La preuve du Théorème 10 est disponible à la Section 6.2.2. □

Implémentation

Pour montrer l'efficacité de notre schéma, nous avons également réalisé une implémentation du protocole `AccessControl` sur une carte SIM NFC classique.

Environnement. La partie "utilisateur" est implémentée sur une carte SIM Javacard 2.2.2, conforme à GlobalPlatform, et embarquée dans un téléphone NFC Samsumg Galaxy S3 qui

fait office de badge. La seule particularité de notre carte est la présence d'API additionnelles, fournies par l'encarteur Oberthur. Ainsi, il est possible de réaliser des opérations modulaires et de l'arithmétique sur les courbes elliptiques. En effet, cette carte SIM dispose d'un crypto-processeur qui lui permet de réaliser des calculs de cryptographie à clé publique, basée sur les courbes elliptiques. Elle est donc plus performante que les cartes SIM classiques utilisées aujourd'hui. Néanmois, ce type de carte est de plus en plus déployées par les opérateurs offrant des services basés sur le NFC.

La partie "borne" de la société est exécutée sur un PC Quad-Core Intel Xeon CPU @3.70GHz. Les communications entre la carte SIM, le téléphone et l'ordinateur sont réalisées par NFC avec un lecteur PC/SC standard, un Omnikey 5321. De cette façon, notre implémentation peut s'adapter à différents cas d'usage.

	Batterie On	**Batterie Off**
(1) Pré-calculs de la carte	(1672 - 1688) 1678	
(2) Transfert : carte $\hookrightarrow \mathcal{S}$	(66-68) 67	85
(3) Calculs par $\mathcal{S}$	(9-34) 22	
(4) Transfert : $\mathcal{S} \hookrightarrow$ carte	(96-115) 102	(175-184) 182
(5) Vérification par la carte	(501-522) 511	
Total en-ligne	(186-224) **205**	(298-322) **315**

TABLE 6.1 – Temps d'exécution ((min-max) moyen) en ms du protocole `AccessControl`

Étude des résultats. L'implémentation utilise une courbe elliptique Barreto-Naehrig [BN06] sur un corps de 256 bits, adaptée à l'utilisation de couplages. Le Tableau 6.1 représente les temps d'exécution obtenus.

Le terme "Batterie Off" désigne un téléphone éteint soit volontairement par l'utilisateur, soit parce que la batterie est déchargée. Dans ce cas, les standards NFC assurent que l'accès NFC à la carte SIM est toujours possible mais avec des performances amoindries. La signature RSA calculée par $\mathcal{S}$ est faite avec un petit exposant public de vérification pour avoir une vérification rapide.

En moyenne, la partie "en ligne" de notre protocole `AccessControl` est rapide, y compris quand le téléphone est éteint. Finalement, ce sont les communications qui représentent l'étape la plus coûteuse.

6.2.2 Preuves de sécurité

Nous prouvons ici le Théorème 10 relatif à la sécurité de notre schéma de paiement électronique privé, selon le modèle présenté à la Section 6.1.2. Pour les preuves par réduction, nous notons $\mathcal{B}$ cette réduction pour éviter toute confusion avec les autorités $\mathcal{R}$.

Nous utilisons ici le lemme de bifurcation (*forking lemma*), introduit par POINTCHEVAL et STERN en 1996 [PS96]. Il permet de montrer la sécurité de schéma de signatures, dans le modèle de l'oracle aléatoire, et utilise la technique dite de rejeu. Ainsi, pour des données identiques en entrée, l'attaquant est amené à fournir deux signatures d'une forme spécifique qui permettent alors de résoudre le problème difficile sous-jacent. Plus précisément, le lemme de bifurcation donne la probabilité d'obtenir deux signatures valides différentes en utilisant le rejeu. Nous renvoyons le lecture à la thèse [Poi96] pour plus de détails.

Preuve de résistance aux falsifications

Soit $\mathcal{A}$ un adversaire contre la propriété de résistance aux falsifications de notre schéma de paiement électronique privé, définie à la Figure 6.1.

La réduction $\mathcal{B}$ utilise l'adversaire $\mathcal{A}$ contre l'hypothèse UF-CMVA du schéma de MAC algébrique $\mathsf{MAC}_{\mathsf{GGM}}$. En entrée, $\mathcal{B}$ utilise les paramètres publics $(q, \mathbb{G}, g, h, C_{x_0}, X_1, X_2)$ fournis par le challenger $\mathcal{C}$ qui sélectionne également la clé secrète (x_0, x_1, x_2) de la société. $\mathcal{B}$ choisit la paire $(sk_{\mathcal{S}}, pk_{\mathcal{S}})$. Les paires de clés ElGamal et Paillier des autorités de révocation peuvent être choisies par $\mathcal{B}$ ou $\mathcal{A}$. Cette réduction a accès aux oracles $\mathcal{O}_{\mathtt{MAC}_{\mathtt{GGM}}}$ et $\mathcal{O}_{\mathtt{Verif}_{\mathtt{GGM}}}$. Le premier génère un $\mathsf{MAC}_{\mathsf{GGM}}$ et le deuxième permet de vérifier la validité. $\mathcal{B}$ fournit l'ensemble des paramètres publics $(q, \mathbb{G}, g, h, C_{x_0}, X_1, X_2)$ à $\mathcal{A}$ et répond à ses requêtes comme suit :

- Requêtes $\mathcal{O}_{\mathtt{Register}}(ID_u)$: étant donné ID_u en entrée, $\mathcal{B}$ choisit aléatoirement sk_u dans $\mathbb{Z}_q^*$, calcule la clé publique associée $pk_u = g^{sk_u}$ et transmet à $\mathcal{A}$;
- Requêtes $\mathcal{O}_{\mathtt{RegisterCorrupt}}(ID_u, (sk_u, pk_u))$: $\mathcal{B}$ n'intervient pour cet oracle ;
- Requêtes $\mathcal{O}_{\mathtt{Corrupt}}(ID_u)$: étant donné ID_u en entrée, $\mathcal{B}$ fournit la clé privée sk_u associée – générée auparavant avec $\mathcal{O}_{\mathtt{Register}}$;
- Requêtes $\mathcal{O}_{\mathtt{TokenIssuance}_{\mathcal{S}}}(pk_u)$: étant donné pk_u, $\mathcal{B}$ utilise la technique de rejeu et d'extraction de la preuve π_1 pour extraire les secrets s et r puis choisit aléatoirement s' dans $\mathbb{Z}_q^*$ avant de faire appel à $\mathcal{O}_{\mathtt{MAC}_{\mathtt{GGM}}}$ sur le message $s_u = s + s'$ pour obtenir une paire (u, u') ; elle transmet $(u, u'' = u'u^r)$ et s' à $\mathcal{A}$ avec une preuve π_2 simulée dans le modèle de l'oracle aléatoire ;
- Requêtes $\mathcal{O}_{\mathtt{AccessControl}_{\mathcal{S}}}(pk_u, T)$: étant donné (pk_u, T), $\mathcal{B}$ utilise la technique de rejeu et d'extraction de la preuve π_3 pour extraire les secrets s_u, m, t, z_1, z_2 et r puis calcule alors le jeton (w, w') correspondant dont elle vérifie la validité avec $\mathcal{O}_{\mathtt{Verif}_{\mathtt{GGM}}}$; si le s_u associé n'a jamais été utilisé pour une requête alors la réduction stoppe puisque (w, w', s_u, m) serait une contrefaçon ; sinon, $\mathcal{B}$ fait appel à $\mathcal{O}_{\mathtt{MAC}_{\mathtt{GGM}}}$ sur $(s_u, m + m')$ pour obtenir $(y, y' = y^{x_0 + x_1 s_u + x_2(m+m')})$ qu'il transmet à $\mathcal{A}$ sous la forme $(y, y'' = y'y^t)$ avec la preuve π_4 simulée et accompagnée d'une signature RSA grâce à $sk_{\mathcal{S}}$;
- Requêtes $\mathcal{O}_{\mathtt{TollComputation}}(pk_u, T, m)$: étant donné pk_u, $\mathcal{B}$ utilise la technique de rejeu et d'extraction de la preuve π_5 pour extraire le secret r et donc le jeton $T = (R, R')$ associé aux valeurs fournies par $\mathcal{A}$; elle utilise $\mathcal{O}_{\mathtt{Verif}_{\mathtt{GGM}}}$ sur le jeton T pour le vérifier puis retourne la valeur totale facturée à $\mathcal{A}$.

Suivant la façon dont procède l'adversaire $\mathcal{A}$, nous distinguons deux types de contrefaçons possibles :

Type 1 : l'adversaire $\mathcal{A}$ réalise le protocole `AccessControl` et la transcription associée $\mathtt{transc}_{\mathtt{AC}}$ ne permet pas de retrouver l'utilisateur correspondant avec $\mathtt{Revocation}_{ID}$;

Type 2 : l'adversaire $\mathcal{A}$ essaye de payer moins que le montant $\tilde{M}$ dû à la société.

Type 1. Le but de l'adversaire $\mathcal{A}$ est de réaliser un contrôle d'accès pour lequel la transcription $\mathtt{transc}_{\mathtt{AC}}$ donne

$$\mathtt{Revocation}_{ID}\left(\mathcal{S}(\mathtt{transc}_{\mathtt{AC}}, \mathtt{DB}_{\mathtt{REG}}), \mathcal{R}(sk_{\mathcal{R}}, pk_{\mathcal{R}})\right) = \perp.$$

Deux cas distincts sont possibles :

Cas 1. Lors de l'appel à l'oracle $\mathcal{O}_{\texttt{AccessControl}_\mathcal{S}}$, le chiffré E' de $g^{s'_u}$ correspond à un jeton T' obtenu avec $\mathcal{O}_{\texttt{TokenIssuance}_\mathcal{S}}$ mais aucune signature σ n'a été fournie. Cela signifie que $\mathcal{A}$ n'a pas reçu s' tel que $s'_u = s + s'$ mais uniquement $g^{s'}$. Or, grâce au lemme de bifurcation et à la propriété de validité de la preuve π_3, $\mathcal{A}$ connaît forcément s'_u – ainsi que s qu'il a lui-même choisi. Il peut donc retrouver s facilement. Étant donné que $\mathcal{O}_{\texttt{TokenIssuance}_\mathcal{S}}$ ne révèle pas le secret s' mais uniquement $g^{s'}$, $\mathcal{A}$ peut être utilisé contre le problème DL. Ce cas est donc négligeable sous l'hypothèse DL.

Cas 2. Le chiffré E de $g^{s'_u}$ ne correspond pas à un jeton obtenu avec $\mathcal{O}_{\texttt{TokenIssuance}_\mathcal{S}}$. s'_u n'a donc pas été utilisé pour une requête de $\mathsf{MAC}_{\mathsf{GGM}}$. Par le lemme de bifurcation et la propriété de validité de π_3, $\mathcal{B}$ extrait les secrets associés à $\texttt{transc}_{\texttt{AC}}$ à savoir (s'_u, m, w, w'). Ainsi, $\mathcal{B}$ est capable de retourner une paire (w, w') qui est un MAC valide sur (s'_u, m) qui n'a pas été obtenu par un appel à $\mathcal{O}_{\texttt{MAC}_{\texttt{GGM}}}$. Par conséquent, $\mathcal{B}$ casse la propriété UF-CMVA du schéma $\mathsf{MAC}_{\mathsf{GGM}}$.

Type 2. Le but de l'adversaire $\mathcal{A}$ est d'obtenir n jetons tels que la somme des valeurs m'_i dues par ces jetons $T_{(i)}$ soit inférieure au montant réel à payer selon les transcriptions dans la base de données $\texttt{DB}_{\texttt{AC}}$, associées à m'_i :

$$\sum_{i=1}^{n'} \texttt{TollComputation}\left(T_{(i)}, m'_i, (sk_\mathcal{S}, pk_\mathcal{S}, \texttt{DB}_{\texttt{REG}})\right) < \sum_{i=1}^{|\texttt{DB}_{\texttt{AC}}|} m_i.$$

Trois cas différents sont possibles :

Cas 1. Ici, $\mathcal{A}$ trouve un moyen d'utiliser un jeton plus de $N_{\texttt{max}}$ fois. Soit $(s_1, \ldots, s_n)$ l'ensemble des secrets utilisés lors des appels à $\mathcal{O}_{\texttt{TokenIssuance}_\mathcal{S}}$ pour obtenir les jetons. Il devrait y avoir $N_{\texttt{max}} \times n$ paires distinctes (s_{u_i}, g_j) pour i allant de 1 à n et j allant de 1 à $N_{\texttt{max}}$. Nous notons L l'ensemble des paires valides. Dans ce cas, il existe au moins une transcription $\texttt{transc}'_{\texttt{AC}}$ dans $\texttt{DB}_{\texttt{AC}}$ dont la paire associée (s'_u, g') n'appartient pas à l'ensemble L. Étant donné que g' appartient forcément à l'ensemble des $\{g_j\}_{j=1}^{N_{\texttt{max}}} - \texttt{transc}'_{\texttt{AC}}$ serait invalide sinon, cela signifie que s_u n'appartient pas à l'ensemble des $\{s_1, \ldots, s_n\}$ et n'a donc pas été utilisé pour une requête de jeton $\mathcal{O}_{\texttt{TokenIssuance}_\mathcal{S}}$ d'où aucun appel à $\mathcal{O}_{\texttt{MAC}_{\texttt{GGM}}}$ avec s'_u. La réduction $\mathcal{B}$ choisit au hasard $\texttt{transc}^{(k)}_{\texttt{AC}}$ dans $\texttt{DB}_{\texttt{AC}}$. La probabilité qu'elle choisisse $\texttt{transc}'_{\texttt{AC}}$ est de $\frac{1}{|\texttt{DB}_{\texttt{AC}}|}$ où $|\texttt{DB}_{\texttt{AC}}|$ correspond au nombre de transcriptions dans $\texttt{DB}_{\texttt{AC}}$. D'après le lemme de bifurcation et la validité de la preuve π_3, $\mathcal{B}$ peut extraire les secrets (s'_u, m, w, w') et retourne alors une paire (w, w') qui est un $\mathsf{MAC}_{\mathsf{GGM}}$ valide pour (s'_u, m) non généré par $\mathcal{O}_{\texttt{MAC}_{\texttt{GGM}}}$. Ainsi, $\mathcal{B}$ casse la propriété UF-CMVA de $\mathsf{MAC}_{\mathsf{GGM}}$.

Cas 2. Au moins une des transcriptions liées à une requête de $\mathcal{A}$ n'a pas été effectuée avec un jeton généré par $\mathcal{O}_{\texttt{TokenIssuance}_\mathcal{S}}$. Ce cas revient à la contrefaçon de type 1, cas 2.

Cas 3. Au moins un des n jetons valides $T^* = (u^*, u'^*)$ fournit par $\mathcal{A}$ n'a pas été généré par $\mathcal{O}_{\texttt{TokenIssuance}_\mathcal{S}}$ ou mis à jour par $\mathcal{O}_{\texttt{AccessControl}}$. Il s'agit donc d'une contrefaçon. Nous posons s^*_u et m^* respectivement le secret et le montant courant associé au jeton T^*. Aucun appel à $\mathcal{O}_{\texttt{MAC}_{\texttt{GGM}}}$ n'a été fait par $\mathcal{B}$ pour la paire (s^*_u, m^*). Ainsi, $\mathcal{B}$ choisit un jeton $T_{(k)}$ parmi l'ensemble des jetons produits par $\mathcal{A}$ avec probabilité $\frac{1}{n}$ de prendre T^*. D'après le lemme de bifurcation et la propriété de validité de π_3, $\mathcal{B}$ peut extraire les secrets associés

à T^* à savoir (s_u^*, m^*, u^*, u'^*) et retourne la paire (u^*, u'^*) qui est un $\mathsf{MAC}_{\mathsf{GGM}}$ valide pour (s_u^*, m^*). $\mathcal{B}$ casse donc la propriété UF-CMVA de $\mathsf{MAC}_{\mathsf{GGM}}$.

Notre système de paiement électronique privé satisfait donc la propriété de résistance aux falsifications, dans le modèle de l'oracle aléatoire, sous l'hypothèse que $\mathsf{MAC}_{\mathsf{GGM}}$ est UF-CMVA.

Preuve de la non-diffamation

Soit $\mathcal{A}$ un adversaire contre la propriété de non-diffamation de notre schéma de paiement électronique privé, définie à la Figure 6.2. Suivant la façon dont procède l'adversaire, deux types de contrefaçons sont possibles :

Type 1 : l'adversaire $\mathcal{A}$ est capable de retrouver le secret s_u associé au jeton généré suite à un appel à $\mathcal{O}_{\texttt{TokenIssuance}_u}$;

Type 2 : l'adversaire $\mathcal{A}$ parvient à générer une signature σ valide à l'insu d'un utilisateur honnête ; il peut donc se faire passer pour lui et obtenir les jetons qu'il souhaite.

La réduction $\mathcal{B}$ utilise l'adversaire $\mathcal{A}$ contre l'hypothèse du logarithme discret DL pour répondre aux challenges fournis par le challenger $\mathcal{C}$. Elle a donc accès à un oracle $\mathcal{O}_{\mathsf{DL}}$. Selon le type d'adversaire, la réduction $\mathcal{B}$ fonctionne différemment. Pour commencer, $\mathcal{B}$ choisit donc aléatoirement un bit c_{mode} qui détermine quel type de contrefaçon sera produite.

Type 1. En entrée, $\mathcal{B}$ utilise les paramètres (g, g^s) fournis par le challenger $\mathcal{C}$ où g est un générateur aléatoire de $\mathbb{G}$. Elle choisit la paire de clé de Paillier $(sk_{\mathcal{R}_p}, pk_{\mathcal{R}_p})$ et sélectionne aléatoirement $\alpha, \beta_1, \ldots, \beta_n$ dans $\mathbb{Z}_p^*$. Elle pose $h = g^\alpha$ et $g_i = g^{\beta_i}$ pour tout $i \in \{1, \ldots, N_{\texttt{max}}\}$. L'adversaire $\mathcal{A}$ choisit la clé privée $\vec{sk} = (x_0, x_1, x_2)$ de la société et la paire ElGamal $(sk_{\mathcal{R}_E}, pk_{\mathcal{R}_E})$. $\mathcal{B}$ fournit les paramètres publics à $\mathcal{A}$ et répond à ses requêtes comme suit :

- Requêtes $\mathcal{O}_{\texttt{Register}}(ID_u)$: étant donné ID_u en entrée, $\mathcal{B}$ choisit aléatoirement sk_u dans $\mathbb{Z}_q^*$, calcule la clé publique associée $pk_u = g^{sk_u}$ et la transmet à $\mathcal{A}$;
- Requêtes $\mathcal{O}_{\texttt{RegisterCorrupt}}(ID_u, (sk_u, pk_u))$: $\mathcal{B}$ n'intervient pas pour cet oracle ;
- Requêtes $\mathcal{O}_{\texttt{Corrupt}}(ID_u)$: étant donné ID_u en entrée, $\mathcal{B}$ fournit la clé privée sk_u associée – générée auparavant avec $\mathcal{O}_{\texttt{Register}}$.
- Requêtes $\mathcal{O}_{\texttt{TokenIssuance}_u}(pk_u)$: étant donné pk_u, $\mathcal{B}$ choisit aléatoirement δ_i et λ_i puis, à partir de son challenge g^s, elle pose $W = g^{\delta_i s + \lambda_i} = g^{s_i}$; pour tout s_i, il existe r, r_1 tels que $C = h^r X_1^{s_i}$ et $D = g_P^{s_i} r_1^n$ et la réduction $\mathcal{B}$ peut simuler π_1 et σ dans le modèle de l'oracle aléatoire ; $\mathcal{B}$ fournit alors à $\mathcal{A}$ un jeton valide tel que $s_{u_i} = s_i + s_i'$ où s_i' est choisi aléatoirement par $\mathcal{A}$ tandis que $s_i = \delta s_i + \lambda_i$ reste inconnu des deux parties ;
- Requêtes $\mathcal{O}_{\texttt{AccessControl}_u}(pk_u, m)$: étant donné (pk_u, m), $\mathcal{B}$ peut simuler la plupart des valeurs nécessaires à l'exécution d'`AccessControl` puisqu'il connaît (u, u') ; pour l, z_1, z_2, r, a dans $\mathbb{Z}_q^*$, $\mathcal{B}$ calcule $c' = (u')^l g^r$, $V = g^{-r} X_1^{z_1} X_2^{z_2}$, $c_2 = w^m h^{z_2}$, $e_1 = g_E^a$ et $e_2 = g^{s_i} g^{s_i'} X_T^a$; pour A, T_i et c_1, $\mathcal{B}$ ne connaît pas toutes les valeurs et génère aléatoirement A puisque pour tout (s_{u_i}, m), il existe un t tel que $A = h^t X_1^{s_{u_i}} X_2^m$; grâce à la propriété de rejeu et de validité de π_2 précédemment, elle extrait b, calcule $c_1 = ((g^{s_i})^\alpha h^{s_i'})^{bl} h^{z_1}$ ainsi que $T_i = (g^{s_i} g^{s_i'})^{\beta_i} = g_i^{s_{u_i}}$ puis simule π_3 et transmet l'ensemble des valeurs à $\mathcal{A}$.

$\mathcal{A}$ retourne une transcription valide $\texttt{transc}'_{\texttt{AC}}$ d'une exécution du protocole `AccessControl` telle que

$$\texttt{Revocation}_{ID}\left(\mathcal{S}(\texttt{transc}'_{\texttt{AC}}, \texttt{DB}_{\texttt{REG}}), \mathcal{R}(sk_{\mathcal{R}}, pk_{\mathcal{R}})\right) = ID_u \in \mathcal{HU}$$

avec une signature de Schnorr σ attestant que le jeton généré pour ID_u a bien été utilisé. Enfin, comme défini par l'expérience de sécurité, $\texttt{transc}'_{\texttt{AC}}$ ne doit pas avoir été produite par un appel à $\mathcal{O}_{\texttt{AccessControl}_u}$.

Ainsi, d'après le lemme de bifurcation, $\mathcal{A}$ est capable de produire une transcription valide $\texttt{transc}'_{\texttt{AC}}$ qui peut générer deux transcriptions valides $\texttt{transc}'_{\texttt{AC}_1}$ et $\texttt{transc}'_{\texttt{AC}_2}$. Ainsi, en utilisant la technique de rejeu et de validité de la preuve π_3, $\mathcal{B}$ peut extraire les valeurs secrètes associées à (s_{u_i}, m, u, u'). Étant donné s_{u_i} et s'_i, $\mathcal{B}$ retrouve $s_i = s_{u_i} - s'_i$. Puisqu'elle connaît les valeurs δ_i et λ_i, elle peut retrouver le logarithme discret s du challenge reçu g^s, ce qui contredit l'hypothèse DL.

Type 2. Ici, pour plus de clarté, nous construisons la réduction $\mathcal{B}$ contre l'hypothèse OMDL. Ainsi, $\mathcal{B}$ utilise en entrée les paramètres $(g^{sk_1}, \ldots, g^{sk_n})$, fournis par le challenger $\mathcal{C}$, où g est un générateur aléatoire de $\mathbb{G}$. $\mathcal{A}$ choisit la clé $\vec{sk} = (x_0, x_1, x_2)$ de la société. Les autres valeurs sont choisies de façon aléatoire et indifféremment par $\mathcal{A}$ $\mathcal{B}$ transmet les paramètres publics à $\mathcal{A}$ et répond à ses requêtes comme suit :

- Requêtes $\mathcal{O}_{\texttt{Register}}(ID_{u_i})$: étant donné ID_{u_i} en entrée, $\mathcal{B}$ choisit un sk_{u_i} de son challenge et transmet $pk_{u_i} = g^{sk_{u_i}}$ à $\mathcal{A}$;
- Requêtes $\mathcal{O}_{\texttt{RegisterCorrupt}}(ID_{u_i}, (sk_{u_i}, pk_{u_i}))$: $\mathcal{B}$ n'intervient pas pour cet oracle ;
- Requêtes $\mathcal{O}_{\texttt{Corrupt}}(pk_{u_i})$: étant donné ID_{u_i} en entrée, $\mathcal{B}$ fait appel à son oracle $\mathcal{O}_{\mathsf{DL}}$ avec pk_{u_i} et transmet la réponse sk_{u_i} à $\mathcal{A}$;
- Requêtes $\mathcal{O}_{\texttt{TokenIssuance}_u}(pk_{u_i})$: étant donné pk_{u_i}, $\mathcal{B}$ choisit aléatoirement s dans $\mathbb{Z}_p^*$ et calcule les valeurs nécessaires à une exécution de `TokenIssuance` ; elle simule la preuve π_1 et la signature σ dans le modèle de l'oracle aléatoire et transmet ces valeurs à $\mathcal{A}$;
- Requêtes $\mathcal{O}_{\texttt{AccessControl}_u}(pk_{u_i}, m)$: étant donné pk_{u_i}, m, $\mathcal{B}$ utilise $(u, u'), s_{u_i}$ construit pour $\mathcal{U}_i$, elle simule les valeurs et les transmet à $\mathcal{A}$.

Après au plus q requêtes à $\mathcal{O}_{\texttt{Corrupt}}$, $\mathcal{A}$ retourne une transcription $\texttt{transc}'_{\texttt{AC}}$ valide, d'une exécution du protocole `AccessControl`. Par définition dans l'expérience de sécurité, l'utilisateur $\mathcal{U}$ associé est honnête et n'a donc pas fait l'objet d'une requête $\mathcal{O}_{\texttt{Corrupt}}$. Ainsi, la clé publique associée $pk_{u_i} = g^{sk_{u_i}}$ fait partie du challenge OMDL. Dans ce cas, l'adversaire $\mathcal{A}$ gagne s'il parvient à générer une signature de Schnorr σ lui permettant de valider un jeton. Or, par le lemme de bifurcation, si l'adversaire est capable de générer une signature σ alors il peut produire deux signatures de Schnorr avec une probabilité non-négligeable. En utilisant la technique de rejeu, $\mathcal{B}$ peut extraire sk_{u_i} utilisé pour générer la signature. Ainsi, $\mathcal{B}$ retourne sk_{u_i} et les q clés secrètes $\{sk_{u_j}\}_{j=1}^{q}$ utilisées pour les requêtes à l'oracle $\mathcal{O}_{\mathsf{DL}}$, ce qui contredit le problème OMDL.

Preuve de la non-chaînabilité

Nous considérons une version plus faible que la non-chaînabilité présentée à la Figure 6.3. Ici, l'adversaire n'a pas accès à l'oracle $\mathcal{O}_{\texttt{Revocation}_{ID}}$. Cette hypothèse est tout à fait réaliste puisque, dans le cas d'une utilisation en pratique, l'accès à cet algorithme est contrôlé.

Pour atteindre la non-chaînabilité parfaite, il faudrait utiliser une version différente du chiffrement ElGamal classique qui n'est que IND-CPA. Or, en chiffrant deux fois un clair à

l'aide d'un chiffrement IND-CPA pour deux clés publiques distinctes, nous avons du IND-CCA2 [NY90]. Par conséquent, il faudrait utiliser du double chiffrement ElGamal, prouvé IND-CCA2 [FP01]. Enfin, nous supposons que les valeurs g_i sont générées à l'aide d'une fonction pseudo aléatoire [DY05]. Ainsi, à chaque contrôle d'accès, l'utilisateur utilise un T_i de la forme $T_i = g^{\frac{1}{s_u+i+1}}$ pour tout i allant de 1 à $N_{\texttt{max}}$.

Soit $\mathcal{A}$ un adversaire contre la propriété de non-chaînabilité de notre système de paiement électronique privé. $\mathcal{A}$ est dit de Type-(i,j) avec i et j inférieurs ou égaux à $N_{\texttt{max}}-1$ si, au cours du challenge, il effectue au plus i appels à $\mathcal{O}_{\texttt{AccessControl}_u}$ sur u_0 et j appels à $\mathcal{O}_{\texttt{AccessControl}_u}$ sur u_1. Ce type Type-(i,j) est un cas particulier de Type-$(N_{\texttt{max}}-1, N_{\texttt{max}}-1)$. Sans perte de généralité, nous nous focalisons donc sur le cas Type-$(N_{\texttt{max}}-1, N_{\texttt{max}}-1)$.

Nous utilisons la technique de preuve par jeux, formalisée par Shoup [Sho04]. Nous décrivons en détail le jeu initial puis plus brièvement les suivants.

Le **jeu 0** correspond à l'attaque réelle, par l'adversaire $\mathcal{A}$. Le challenger $\mathcal{C}$ choisit aléatoirement les paramètres publics du système $pp = (q, \mathbb{G}, g, h, g_R, N_{\texttt{max}}, \{g_i\}_{i=0}^{N_{\texttt{max}}})$, les clés $\vec{sk} = (x_0, x_1, x_2)$ et $sk_{\mathcal{S}}$ de la société ainsi que les paires de clés $\big((sk_{\mathcal{R}_E}, pk_{\mathcal{R}_E}), (sk_{\mathcal{R}_P}, pk_{\mathcal{R}_P})\big)$. Il transmet $\vec{sk}$ et $sk_{\mathcal{S}}$ à $\mathcal{A}$ puis répond à ses requêtes comme suit :

- Requêtes $\mathcal{O}_{\texttt{Register}}(ID_u)$: étant donné ID_u en entrée, $\mathcal{C}$ choisit aléatoirement sk_u dans $\mathbb{Z}_q^*$ et calcule la clé publique $pk_u = g^{sk_u}$ qu'il transmet à $\mathcal{A}$;
- Requêtes $\mathcal{O}_{\texttt{RegisterCorrupt}}(ID_u)$: $\mathcal{C}$ n'intervient pas pour cet oracle ;
- Requêtes $\mathcal{O}_{\texttt{Corrupt}}(ID_u)$: étant donné ID_u en entrée, $\mathcal{C}$ fournit la clé privée sk_u associée ;
- Requêtes $\mathcal{O}_{\texttt{TokenIssuance}_u}(pk_u)$: étant donné pk_u, $\mathcal{C}$ choisit une valeur aléatoire s dans $\mathbb{Z}_q^*$, obtient un jeton sur $s_u = s + s'$ et génère une signature de Schnorr σ à partir de sk_u ;
- Requêtes $\mathcal{O}_{\texttt{AccessControl}_u}(pk_u, m)$: étant donné pk_u et m, $\mathcal{C}$ utilise le jeton (u, u'), le secret s_u construits pour $\mathcal{U}$ et g_i pour réaliser un `AccessControl` correspondant au montant m ;
- Requêtes $\mathcal{O}_{\texttt{TollComputation}}(pk_u, T, m)$: étant donné pk_u, T et m, $\mathcal{C}$ vérifie et retourne la valeur totale facturée à $\mathcal{A}$.

$\mathcal{A}$ sélectionne deux utilisateurs honnêtes u_0 et u_1 ainsi que deux montants m_0 et m_1 qu'il retourne à $\mathcal{C}$. Pour u_0 et u_1, $\mathcal{C}$ réalise le protocole `TokenIssuance` et obtient les transcriptions $\texttt{transc}_{\texttt{AC}}^{(0)}$ et $\texttt{transc}_{\texttt{AC}}^{(1)}$ qu'il transmet à $\mathcal{A}$. Pour un bit b aléatoire, choisi par $\mathcal{C}$, il exécute le protocole `AccessControl` avec respectivement $(T_{(1)}, m_b)$ et $(T_{(0)}, m_{1-b})$. Les transcriptions obtenues $\texttt{transc}_{\texttt{AC}}^{(b)}$ et $\texttt{transc}_{\texttt{AC}}^{(1-b)}$ sont envoyées à $\mathcal{A}$. Son but est de déterminer la valeur du bit b. Autrement dit, il cherche à identifier quel utilisateur, u_0 ou u_1, a exécuté `AccessControl` avec le montant m_0 ou m_1. Après avoir reçu ce challenge, $\mathcal{A}$ a toujours accès aux différents oracles mais avec des restrictions pour $\mathcal{O}_{\texttt{TollComputation}}$ comme nous l'avions défini avec l'expérience. Il ne peut pas corrompre les utilisateurs cibles ou tenter de révoquer– l'anonymat ou les jetons. Dans cette phase, les montants pour u_0 et u_1 doivent être égaux, tout comme leur nombres d'accès, sinon, $\mathcal{A}$ remporterait trivialement ce jeu.

À la fin du jeu, $\mathcal{A}$ retourne un bit b'. Pour S_0 l'évènement "$b' = b$", nous avons alors :

$$\texttt{Adv}_{\mathcal{A}}^{\text{NC}-b}(1^\lambda) = |\mathbb{P}(\texttt{Exp}_{\mathcal{A}}^{\text{NC}-b}(1^\lambda) = b) - 1/2| = |\mathbb{P}(S_0) - 1/2|.$$

Pour chaque jeu i, cet évènement est noté S_i.

Afin d'éviter que l'adversaire $\mathcal{A}$ puisse détecter les différents changements entre deux jeux successifs, nous construisons également les jeux suivants :

Jeu(0,b) - *Transition basée sur l'indistinguabilité du chiffrement ElGamal et* DDH.

Le jeu $(0,b)$ est identique au jeu 0, excepté pour le chiffré ElGamal durant l'exécution de `AccessControl` pour u_b, noté E_b. Le challenger $\mathcal{C}$ choisit une valeur aléatoire pour E_b et simule la preuve $\pi_3^{(b)}$ dans le modèle de l'oracle. Sous l'hypothèse DDH, l'adversaire $\mathcal{A}$ ne peut pas distinguer ce changement. En effet, il est possible de créer un distingueur $\mathcal{D}$ avec un avantage $\mathtt{Adv}_{\mathcal{D}}^{\mathsf{DDH}}(1^\lambda)$ pour le problème DDH. Nous avons alors :

$$|\mathbb{P}(S_0) - \mathbb{P}(S_{(0,b)})| \leq \mathtt{Adv}_{\mathcal{D}}^{\mathsf{DDH}}(1^\lambda).$$

Jeu(0,$1-b$) - *Transition basée sur l'indistinguabilité du chiffrement ElGamal et* DDH.

Le jeu $(0,1-b)$ est identique au jeu $(0,b)$, excepté pour le chiffré ElGamal durant l'exécution de `AccessControl` pour u_{b-1}, noté E_{1-b}. Le challenger $\mathcal{C}$ choisit une valeur aléatoire pour E_{1-b} et simule la preuve $\pi_3^{(1-b)}$ dans le modèle de l'oracle. Sous l'hypothèse DDH, l'adversaire $\mathcal{A}$ ne peut pas distinguer ce changement. En effet, il est possible de créer un distingueur $\mathcal{D}$ avec un avantage $\mathtt{Adv}_{\mathcal{D}}^{\mathsf{DDH}}(1^\lambda)$ pour le problème DDH. Nous avons alors :

$$|\mathbb{P}(S_{(0,b)}) - \mathbb{P}(S_{(0,1-b)})| \leq \mathtt{Adv}_{\mathcal{D}}^{\mathsf{DDH}}(1^\lambda).$$

Jeu(0,$1-b$,b) - *Transition basée sur l'hypothèse* $q-$DDHI.

Le jeu $(0,1-b,b)$ est identique au jeu $(0,1-b)$, excepté pour la valeur $T_i^{(b)} = \frac{1}{g^{s_{u_b}+i+1}}$ lors de l'exécution de `AccessControl` pour u_b que le challenger remplace par une valeur aléatoire. Il simule ensuite la preuve $\pi_3^{(b)}$ dans le modèle de l'oracle aléatoire. Sous l'hypothèse $q-$DDHI, l'adversaire $\mathcal{A}$ ne peut pas distinguer ce changement. En effet, il est possible de créer un distingueur $\mathcal{D}$ avec un avantage $\mathtt{Adv}_{\mathcal{D}}^{q-\mathsf{DDHI}}(1^\lambda)$ pour le problème $q-$DDHI. Nous avons alors :

$$|\mathbb{P}(S_{(0,1-b)}) - \mathbb{P}(S_{(0,1-b,b)})| \leq \mathtt{Adv}_{\mathcal{D}}^{q-\mathsf{DDHI}}(1^\lambda).$$

Jeu(0,$1-b$,$1-b$) - *Transition basée sur l'hypothèse* $q-$DDHI.

Le jeu $(0,1-b,1-b)$ est identique au jeu $(0,1-b,b)$, excepté pour la valeur $T_i^{(1-b)}$ lors de l'exécution de `AccessControl` pour u_{1-b} que le challenger remplace par une valeur aléatoire. Il simule ensuite la preuve $\pi_3^{(1-b)}$ dans le modèle de l'oracle aléatoire. Sous l'hypothèse $q-$DDHI, l'adversaire $\mathcal{A}$ ne peut pas distinguer ce changement. En effet, il est possible de créer un distingueur $\mathcal{D}$ avec un avantage $\mathtt{Adv}_{\mathcal{D}}^{q-\mathsf{DDHI}}(1^\lambda)$ pour le problème $q-$DDHI. Nous avons alors :

$$|\mathbb{P}(S_{(0,1-b,b)}) - \mathbb{P}(S_{(0,1-b,1-b)})| \leq \mathtt{Adv}_{\mathcal{D}}^{q-\mathsf{DDHI}}(1^\lambda).$$

Nous définissons le **jeu 1** comme étant identique au jeu $(0,1-b,1-b)$. D'où l'inégalité suivante :

$$|\mathbb{P}(S_1) - \mathbb{P}(S_0)| \leq 2 \times (\mathtt{Adv}_{\mathcal{D}}^{\mathsf{DDH}}(1^\lambda) + \mathtt{Adv}_{\mathcal{D}}^{q-\mathsf{DDHI}}(1^\lambda)).$$

Nous procédons de la même façon pour tout appel de $\mathcal{A}$ à l'oracle $\mathcal{O}_{\mathtt{AccessControl}_u}$ avec pk_{u_0} et pk_{u_1}. Ainsi, pour k allant de 1 à $N_{\mathtt{max}} - 1$, nous avons les jeux suivants :

Jeu(k,b). Le jeu (k,b) est identique au jeu k qui est équivalent au jeu $(k-1,1-b,1-b)$ excepté pour le calcul du chiffré ElGamal $E_b^{(k)}$ au cours de l'exécution de `AccessControl` pour u_b. $E_b^{(k)}$ est remplacé par une valeur aléatoire et la preuve $\pi_3^{(b,k)}$ est simulée.

Jeu(k,$1-b$). Le jeu $(k,1-b)$ est identique au jeu (k,b) excepté pour le calcul du chiffré ElGamal $E_{1-b}^{(k)}$ au cours de l'exécution de `AccessControl` pour u_{1-b}. $E_{1-b}^{(k)}$ est remplacé par une valeur aléatoire et la preuve $\pi_3^{(1-b,k)}$ est simulée.

Jeu(k,$1-b$,b). Le jeu $(k,1-b,b)$ est identique au jeu $(k-1,1-b,1-b)$ excepté pour $T_i^{(b,k)}$ calculé au cours de l'exécution de `AccessControl` pour u_b. $T_i^{(b,k)}$ est remplacé par une valeur aléatoire et la preuve $\pi_3^{(b,k)}$ est simulée.

Jeu(k,$1-b$,$1-b$). Le jeu $(k,1-b,1-b)$ est identique au jeu $(k,1-b,b)$ excepté pour $T_i^{(b,k)}$ calculé au cours de l'exécution de `AccessControl` pour u_{1-b}. $T_i^{(1-b,k)}$ est remplacé par une valeur aléatoire et la preuve $\pi_3^{(1-b,k)}$ est simulée.

Nous posons alors que le **jeu $k+1$** est identique au jeu $(k,1-b,1-b)$. D'où l'inégalité suivante :

$$|\mathbb{P}(S_{k+1}) - \mathbb{P}(S_k)| \leq 2 \times (\mathtt{Adv}_{\mathcal{D}}^{\mathsf{DDH}}(1^\lambda) + \mathtt{Adv}_{\mathcal{D}}^{q-\mathsf{DDHI}}(1^\lambda)).$$

En poursuivant avec le même raisonnement, nous définissons le jeu $N_{\mathtt{max}}-1$ comme étant identique au jeu $(N_{\mathtt{max}}-2,1-b,1-b)$. Cependant, dans le dernier jeu, $\mathcal{C}$ ne donne aucune information sur le bit b à l'adversaire $\mathcal{A}$ au sens de la théorie de l'information. En effet, cela est dû au fait que toutes les valeurs utilisant s_u ont été remplacées par des valeurs aléatoires et sont donc parfaitement indistinguables, contrairement aux chiffrés E ou aux valeurs T_i. Ainsi :

$$\mathbb{P}(S_{N_{\mathtt{max}}-1}) = 1/2.$$

Ainsi,

$$\begin{aligned} \mathtt{Adv}_{\mathcal{A}}^{\mathrm{NC}-b}(1^\lambda) &= |\mathbb{P}(\mathtt{Exp}_{\mathcal{A}}^{\mathrm{NC}-b} = b) - 1/2| \\ &= |\mathbb{P}(S_0) - \mathbb{P}(S_{N_{\mathtt{max}}-1})|. \end{aligned}$$

Par conséquent, l'avantage de l'adversaire $\mathcal{A}$ pour la propriété de non-chaînabilité est borné comme suit :

$$\begin{aligned} \mathtt{Adv}_{\mathcal{A}}^{\mathrm{NC}-b}(1^\lambda) &\leq \sum_{j=0}^{N_{\mathtt{max}}-1} |\mathbb{P}(S_j) - \mathbb{P}(S_{j+1})| \\ &\leq 2 \times N_{\mathtt{max}} \times \left(\mathtt{Adv}_{\mathcal{D}}^{\mathsf{DDH}}(1^\lambda) + \mathtt{Adv}_{\mathcal{D}}^{q-\mathsf{DDHI}}(1^\lambda)\right) \end{aligned}$$

Ainsi sous les hypothèses DDH, $q-$DDHI et DCR, l'avantage d'un adversaire de Type-$(N_{\mathtt{max}}-1,N_{\mathtt{max}}-1)$ est négligeable. Il en est donc de même pour tout adversaire de type-(i,j) pour i et j inférieurs ou égaux à $N_{\mathtt{max}}-1$.

Notre système de paiement électronique privé satisfait donc la propriété de non-chaînabilité, dans le modèle de l'oracle aléatoire, sous les hypothèses DDH, $q-$DDHI et DCR.

Conclusion

Dans ce chapitre, nous avons présenté un nouveau système de post-paiement électronique privé. L'utilisateur n'a besoin que d'un unique jeton de paiement qu'il peut utiliser plusieurs fois, sans que cela ne porte atteinte à sa vie privée. Ce jeton est basé sur le MAC algébrique $\mathsf{MAC}_{\mathsf{GGM}}$ et les montants payés sont agrégés grâce à notre signature partiellement aveugle. Ce système est prouvé sûr dans le modèle de l'oracle aléatoire et repose sur différentes hypothèses calculatoires et la sécurité de $\mathsf{MAC}_{\mathsf{GGM}}$. Une implémentation sur carte SIM NFC atteste de son efficacité pour des cas d'usages réels sur périphériques aux capacités limitées en mémoire et en puissance de calcul.

Troisième partie

Anonymisation de données

Chapitre 7

Anonymisation de graphes par la confidentialité différentielle

Dans ce chapitre, nous introduisons un nouveau mécanisme générique d'anonymisation pour les graphes qui assure la confidentialité différentielle. Contrairement aux techniques de la littérature qui ajoutent un bruit uniforme sur l'ensemble des données, notre mécanisme optimise la façon de distribuer le bruit en considérant différents sous-ensembles parmi les arêtes du graphe. En conduisant des expérimentations sur des données réelles, nous comparons notre schéma avec les mécanismes existants pour montrer que l'anonymisation obtenue altère moins les données initiales et offre ainsi une meilleure utilité.

Ce résultat a été publié à la conférence PST 2016, dans l'article intitulé *"Edge-Calibrated Noise for Differentially Private Mechanisms on Graphs"* [BCGO16].

Sommaire

7.1 Confidentialité différentielle et graphes

De plus en plus de données personnelles sont aujourd'hui collectées et stockées, par nécessité de service ou bien par obligation légale. Il peut s'agir par exemple de données médicales, de données extraites à partir des réseaux sociaux ou des réseaux de télécommunications. Celles-ci représentent une grande richesse mais implique également des risques pour la vie privée des individus concernés. À partir de données de mobilité, il est ainsi possible d'inférer des informations personnelles comme le domicile d'un individu, son lieu de travail mais aussi des informations dites sensibles comme ses opinions politiques, religieuses ou d'éventuels problèmes

de santé. L'anonymisation a pour but de supprimer le lien entre l'individu et les données qui le concernent. Au niveau légal, une fois anonymisées, ces bases de données ne sont plus considérées comme étant à caractère personnel et peuvent être publiées à des fins d'études par exemple.

La confidentialité différentielle [DMNS06, Dwo06] (voir Section 2.3.3) est une méthode d'anonymisation très étudiée aujourd'hui. L'avantage de cette technique est qu'elle permet de quantifier formellement le niveau d'anonymat obtenu. Il alors est possible de contrôler le compromis, nécessaire à toute technique d'anonymisation, entre l'utilité des données et le niveau de respect de la vie privée résultant. Plus le bruit ajouté est important, plus le niveau de respect de la vie privée est fort ce qui entraîne une perte d'information importante en contrepartie. À l'inverse, plus le bruit ajouté est faible, plus le risque de ré-identifier les individus est grand. Le paramètre associé à la confidentialité différentielle est donc un réel atout pour jauger l'anonymat assuré. Initialement, cette méthode d'anonymisation a été introduite dans le cas de requêtes statistiques sur une base de données.

Dans cette section, nous introduisons les différentes notations propres à ce chapitre autour des définitions de confidentialité différentielle pour des graphes et matrices. Nous présentons ensuite l'état de l'art principal pour la confidentialité différentielle dans ce contexte.

7.1.1 Définitions et notations

Dans les chapitres introductifs, nous avons présenté la confidentialité différentielle dans le cas plus classique d'une requête simple sur une base de données. Ici, nous présentons brièvement la relation entre graphes et matrices. Nous élargissons ensuite les définitions de la Section 2.3.3 au cas particulier des matrices qui agrègent des données personnelles de plusieurs individus.

Graphes et matrices

Un *graphe* G est un couple (V, E) où V est l'ensemble des sommets de G et E est l'ensemble de ses arêtes. Un graphe $G = (V, E)$ est orienté si chaque arête (u, v) est décrite par un couple ordonné de sommets u et v.En outre, un graphe est pondéré lorsque chaque arête est affectée d'un nombre réel positif appelé *poids* de cette arête.

Pour représenter un graphe G à n sommets, nous considérons sa *matrice d'adjacence* A dans $\mathbb{R}^{n\times n}$. Il s'agit une matrice carrée où lignes et colonnes identifient les sommets du graphes. Le coefficient A_{ij} de la matrice A correspond au poids de l'arête (i, j) c'est-à-dire entre les sommets i et j. Par la suite, par simplicité, nous considérons la représentation sous forme matricielle du graphe pour l'élaboration de notre modèle.

La figure 7.1 illustre un graphe dont la matrice d'adjacence est $A = \begin{pmatrix} 34 & 14 & 3 \\ 0 & 57 & 9 \\ 7 & 8 & 26 \end{pmatrix}$.

Matrices et confidentialité différentielle

Scénario. Pour notre schéma, nous considérons le cas d'un responsable de traitement qui détient un ensemble $\mathcal{D}$ de bases de données. Son but est de publier une version anonymisée de la matrice A, obtenue par $\psi(D)$, qui satisfait la confidentialité différentielle, pour ψ une requête retournant une matrice et D dans $\mathcal{D}$. Nous cherchons donc à construire un mécanisme $\mathcal{M}$:

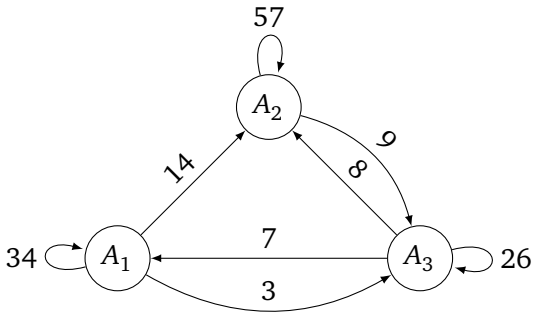


FIGURE 7.1 – Graphe G associé à la matrice d'adjacence A

$\mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ qui, pour une matrice A, retourne une matrice $\tilde{A}$ qui satisfait la confidentialité différentielle. Nous spécifions maintenant les définitions introduites à la Section 2.3.3 dans le cas particulier des matrices.

Matrices voisines et sensibilité. Deux bases de données D et D' dans un ensemble $\mathcal{D}$ sont dites *voisines* si elles diffèrent d'au plus un enregistrement. Cette relation est notée $D \sim D'$. De même, deux matrices A et A' dans $\mathbb{R}^{n\times n}$ sont dits voisines, de la même façon $A \sim A'$, si elles correspondent à deux bases de données voisines l'une de l'autre c'est-à-dire $A = \psi(D)$ et $A' = \psi(D')$ pour $D \sim D'$.

En outre, pour définir la sensibilité d'une matrice, nous utilisons la norme L_1 ou la norme L_2 selon le mécanisme utilisé – Laplace ou Gauss, par exemple. Ainsi, pour toutes matrices A et A' voisines, les sensibilités Δ^{L_1} et Δ^{L_2} sont définies comme suit :

$$\Delta^{L_1} = \max_{A\sim A'} \sum_{i,j} |A_{ij} - A'_{ij}| \quad \text{et} \quad \Delta^{L_2} = \max_{A\sim A'} \sqrt{\sum_{i,j} |A_{ij} - A'_{ij}|^2}.$$

Confidentialité différentielle pour les matrices. Soit $\mathcal{M} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ un mécanisme d'assainissement. Il satisfait la confidentialité différentielle pour ε strictement positif si, pour toutes matrices voisines A et A', pour tout sortie R dans $\mathbb{R}^{n\times n}$,

$$\frac{\mathbb{P}(\mathcal{M}(A) \in R)}{\mathbb{P}(\mathcal{M}(\tilde{A}) \in R)} \leq \exp(\varepsilon).$$

Par soucis de clarté, nous définissons nos mécanismes pour une matrice A, sans préciser systématiquement qu'elle est obtenue à partir d'une requête ψ sur une base de données $D \in \mathcal{D}$.

Pour une requête sur des matrices, les mécanismes classiques peuvent alors s'écrire comme suit. Soit une matrice $A \in \mathbb{R}^{n\times n}$. Le mécanisme $\mathcal{M}$ est défini par

$$\mathcal{M}(A) = A + B = \tilde{A}$$

avec B une matrice "bruit" dont les coefficients sont des variables aléatoires indépendantes et identiquement distribuées selon un paramètre dépendant de la sensibilité de la matrice A, du paramètre ε et, le cas échéant, de la valeur δ pour une version relâchée de la confidentialité différentielle.

Selon le bruit choisi, les variables aléatoires de B sont calibrées différemment avec
- le paramètre $\lambda = \frac{\Delta^{L_1}}{\varepsilon}$ pour le mécanisme de Laplace et la confidentialité différentielle selon ε,
- un écart-type $\sigma = \frac{\Delta^{L_2} \times \sqrt{2\ln(\frac{1{,}25}{\delta})}}{\varepsilon}$ pour le mécanisme de Gauss qui assure alors la confidentialité différentielle pour ε et δ.

Ainsi, le bruit appliqué à la matrice est uniforme pour tous les coefficients (i, j). Dans ce chapitre, notre objectif est d'optimiser ce bruit pour assurer la confidentialité différentielle tout en préservant au mieux l'utilité des données.

7.1.2 État de l'art

De nombreux travaux étudient les techniques permettant d'obtenir des graphes qui respectent la confidentialité différentielle [AJL13, BDMN05, KRSY11, MW09, WW13]. La plupart des travaux précédents ont pour objectif de produire des statistiques sur des graphes – le nombre de sommets, par exemple – respectant la confidentialité différentielle puis de construire des graphes synthétiques, anonymes, à partir de statistiques pertinentes qui ont été perturbées. Cette approche nécessite un échantillonage selon le modèle de Kronecker [MW12, KRSY11] ou encore le modèle exponentiel [KSK14, LM14] pour générer aléatoirement une matrice offrant des garanties fortes de respect de la vie privée. Cependant, utiliser un échantillon n'est pas toujours adapté à toutes les situations concrètes, comme celle de la mobilité par exemple.

En 2013, WANG, WU et WU [WWW13] proposent un mécanisme pour publier des vecteurs et valeurs propres d'une matrice d'adjacence en respectant la confidentialité différentielle. La même année, HARDT et ROTH [HR13] ainsi que KAPRALOV et TALWAR [KT13] introduisent des bornes théoriques pour ces valeurs et la confidentialité différentielle.

Cependant, toutes ces solutions impliquent un manque de contrôle sur la notion de voisinage choisie. Dans notre travaux, nous privilégions la décomposition en valeurs singulières (SVD, *Singular Value Decomposition*) qui permet de factoriser certaines matrices. En 2005, BLUM, DWORK, MCSHERRY et NISSIM [BDMN05] proposent pour la première fois l'utilisation de la SVD pour la confidentialité différentielle. Cette technique a ensuite été largement utilisée [CSS12, DTTZ14], en particulier pour des systèmes de recommandation [MM09]. En effet, MCSHERRY et MIRONOV prouvent en 2009 que l'utilisation du mécanisme de Laplace et d'une approximation de rang k permet de fournir des services de recommandations privés, au sens de la confidentialité différentielle, avec une précision raisonnable.

Enfin, le travail antérieur qui se rapproche le plus de notre contribution est celui de LI, HAY, RASTOGI, MIKLAU et MCGREGOR [LHR+10]. En 2010, ils sont les premiers à essayer d'optimiser l'utilisation du budget de vie privée ε et cherchent alors à résoudre un programme semi-défini avec contraintes de rang. Cette approche est complexe et la notion de voisinage considérée reste restreinte.

Comparaison des différentes notions de vie privée

La confidentialité différentielle dépend, entre autres, de la notion de voisinage considérée. Dans le cas des graphes, plusieurs distinctions existent suivant les données où la vie privée doit être assurée.

Arêtes d'un graphe. La notion la plus étudiée est la confidentialité différentielle appliquée aux arêtes [KRSY11]. Dans ce cas, deux graphes sont dits *voisins* s'ils diffèrent d'une seule arête. Ici, un individu est donc représenté par une seule arrête.

Nœuds d'un graphe. Dans ce cas, deux graphes sont dits voisins s'ils diffèrent d'un seul sommet [KNRS13a]. Ici, l'individu peut alors être vu comme un unique sommet qui agit seulement sur le poids des arêtes auxquelles il est connecté.

Ligne d'une matrice. Deux matrices sont voisines si elles diffèrent d'une simple ligne [DTTZ14]. Dans ce cas, chaque ligne de la matrice ne peut être perturbée que par un seul et unique individu.

Poids des arêtes. La définition la plus proche de celle que nous utilisons est celle introduite dans [Sea16] avec une confidentialité différentielle par rapport aux poids des arrêtes. Dans ce cas, un même individu peut impacter plusieurs arêtes.

7.2 Notre procédé de confidentialité différentielle par blocs

7.2.1 Description

Notre méthode d'anonymisation de graphes basé sur la confidentialité différentielle comprend trois étapes principales : l'identification de l'ensemble des données sensibles, le partitionnement de celles-ci en sous-ensembles et l'évaluation de leurs sensibilités puis l'application du mécanisme de bruitage. Une dernière étape de décomposition en valeurs singulières peut également être appliquée avec ce mécanisme. Il peut être utilisé avec toutes données personnelles qui se représentent sous forme de graphes ou de matrices, en particulier lorsqu'il s'agit de données agrégées correspondant à plusieurs individus comme des données de réseaux sociaux ou encore des traces de navigation sur Internet.

Nous présentons chacune de ces phases dans le cas de l'ajout d'un bruit laplacien puis illustrons leur fonctionnement à travers un exemple concret autour de la mobilité des individus. Les différents théorèmes sont prouvés dans la Section suivante 7.3.2

Initialisation

Soit $\mathcal{D}$ l'espace des bases de données possibles. Pour une base de données $D \in \mathcal{D}$, nous considérons une requête $\psi : \mathcal{D} \to \mathbb{R}^{n \times n}$ qui retourne une matrice carrée A. Cette matrice correspond alors à une agrégation de données d'un certain nombre d'individus.

Exemple des comptes-rendus d'appels. Nous prenons le cas particulier d'un ensemble de base de données $\mathcal{D}$ qui contiennent des traces de mobilité générées à partir des comptes rendus d'appels (CRA) détenus par un opérateur téléphonique. Chaque base de données D dans $\mathcal{D}$ contient alors des données personnelles relatives au comportement des utilisateurs : date, heure et localisation de l'antenne pour chaque appel/SMS émis ou reçu au cours d'une période donnée. Nous supposons que le réseau téléphonique contient n antennes.

Le responsable de traitement peut retrouver les données $D_{u_1}, \ldots, D_{u_N}$ pour les N individus $u_1, \ldots, u_N$ et construire des graphes de mobilité correspondants. Pour chaque paire d'antennes (i, j) qui correspond à une transition entre les antennes i et j, il est alors

possible de compter l'ensemble des appels/sms effectués à l'antenne i suivis d'une autre communication à l'antenne j au cours de la période. Cela correspond à une requête, notée ψ, qui retourne une matrice A contenant l'ensemble des comptages. Plus précisément, pour l'utilisateur u_k ayant pour données D_{u_k} dans D, le nombre de communications de l'antenne i vers l'antenne j est :

$$\psi_{ij}(D_{u_k}) = A_{ij}^{u_k} \text{ dans } \mathbb{N}.$$

Pour chaque transition (i, j) possible, avec i et j allant de 1 à n, le coefficient A_{ij} de la matrice A est l'agrégation des comptages pour l'ensemble des individus :

$$A_{ij} = \sum_{k=1}^{N} A_{ij}^{u_k}.$$

Identification des données sensibles

En pratique, il est courant que certains coefficients de la matrice obtenue soient plus sensibles que d'autres. Ici, le sens de "données sensibles" correspond à celles qui sont susceptibles d'être affectées par l'ajout ou le retrait d'un individu dans la base initiale. Pour préserver au mieux les données et la précision de la sortie, notre objectif est de n'affecter que les coefficients sensibles. En effet, ceux qui sont non-sensibles n'ont pas d'impact sur la vie privée des utilisateurs.

Pour deux matrices voisines A et A' obtenues avec $\psi(\mathcal{D})$, l'ensemble $\mathcal{S}$ des données sensibles est défini par :

$$\mathcal{S} = \{(i, j) \mid A_{ij} \neq A'_{ij} \text{ pour toute matrice } A \sim A'\}.$$

Pour un mécanisme $\mathcal{M}$ qui satisfait la confidentialité différentielle, le fait de l'appliquer uniquement à $\mathcal{S}$ respecte toujours la confidentialité différentielle. En effet, les coefficients qui varient selon la présence ou l'absence d'un individu sont bien perturbés, comme l'impose cette définition. Par défaut, les valeurs $A_{ij} = A'_{ij}$ sont considérées non-sensibles.

Exemple des comptes-rendus d'appels. Dans notre exemple de données de mobilité, les coefficients A_{ij} non-sensibles correspondent aux cas où aucune transition n'a été observée entre les antennes i et j c'est-à-dire $A_{ij} = 0$. Dans ce cas, la matrice obtenue – et, par conséquent, le graphe – est éparse. L'ensemble $\mathcal{S}$ des données sensibles correspond alors aux trajets effectués par au moins un utilisateur, nous avons :

$$\mathcal{S} = \{(i, j) \mid A_{ij}^{u} \neq 0 \text{ pour au moins un utilisateur } u\}.$$

Partitionnement des coefficients sensibles

À partir de l'ensemble $\mathcal{S}$ obtenu auparavant, nous construisons une partition P composée de plusieurs sous-ensembles $S_1, \ldots, S_l$ pour un entier positif l. La partition choisie va ensuite avoir une influence sur la qualité du mécanisme car elle impacte directement le bruit ajouté. En effet, l'objectif de la méthode qui suit est de trouver une séparation des arêtes du graphes en fonction de leur niveau de sensibilité. Ainsi, plus le niveau de sensibilité du sous-ensemble considéré est bas, moins nous y ajouterons du bruit.

Nous supposons ici que le responsable de traitement a connaissance des sensibilités Δ_{ij} de tous les comptages sur les transitions (si ce n'est pas le cas, voir Section 7.2.2 sur la méthode à appliquer). Pour chaque transition (i, j), la sensibilité Δ_{ij} des coefficients de la matrice A est évaluée et définie par :

$$\Delta_{ij} = \max_{A \sim A'} |A_{ij} - A'_{ij}|.$$

Ensuite, dans le cas d'une partition en deux sous-ensembles par exemple, les sensibilités Δ_{ij} de chaque couple sensible (i, j) sont comparées à un seuil fixé τ strictement positif. La partition $P_\tau = (S_1, S_2)$ est alors obtenue comme suit :

$$\begin{aligned} S_1 &= \{(i,j) \mid \Delta_{ij} \leq \tau\}, \\ S_2 &= \{(i,j) \mid \Delta_{ij} > \tau\}. \end{aligned}$$

Chaque sous-ensemble S_k a alors sa propre sensibilité Δ_k qui dépend des données sensibles qu'il contient. Nous définissons alors la sensibilité Δ_k, appelée sensibilité par blocs.

Définition 41 (Sensibilité par blocs pour les matrices)**.** Soit $(S_k)_{1 \leq k \leq K}$ une partition. Pour deux matrices voisines A et A', la *sensibilité par blocs* Δ_k de chaque sous-ensemble S_k est :

$$\Delta_k = \max_{A \sim A'} \sum_{(i,j) \in S_k} |A_{ij} - A'_{ij}|.$$

Exemple des comptes-rendus d'appels. Dans notre exemple, les arêtes les plus sensibles du sous-ensemble S_2 vont correspondre à des lieux particuliers pour l'utilisateur comme son domicile ou son lieu de travail par exemple. En revanche, les lieux très fréquentés par de nombreux utilisateurs vont être moins sensibles pour la vie privée d'un individu précis comme dans le cas d'un aéroport ou encore d'un centre commercial.

Choix de la partition. Le choix du seuil τ impacte la qualité de l'assainissement obtenu. Ainsi, pour deux seuils distincts τ_1 et τ_2, il est facile de comparer les partitions associées P_{τ_1} et P_{τ_2} en regardant les sorties obtenues. La meilleure partition est celle qui minimise l'erreur. En effet, la perte d'utilité induite par l'ajout du bruit peut se mesurer en fonction d'erreur qui est calculée entre la matrice initiale et celle obtenue après application du mécanisme.

Dans le cas d'un bruit laplacien avec une partition en deux sous-ensembles (S_1, n_1, Δ_1) et (S_2, n_2, Δ_2) avec n_i le nombre d'éléments de S_i et Δ_i sa sensibilité, la fonction F à minimiser peut être définie comme l'erreur moyenne L_1 :

$$F(n_1, n_2, \Delta_1, \Delta_2) = \sqrt{n_1 \Delta_1} + \sqrt{n_2 \Delta_2}.$$

Pour illustrer la recherche automatique de la meilleure partition P_τ pour le seuil τ, nous proposons l'algorithme de recherche exhaustive suivant.

Recherche d'une partition en deux sous-ensembles

Entrée : les seuils possibles $\tau_1, \tau_2, \ldots, \tau_r$ dans $[0, \max \Delta_{ij}]$, la fonction d'erreur F à minimiser

Sortie : Indice i de la meilleure partition P_{τ_i} pour le mécanisme par blocs

1. Pour toute paire (i, j), calculer la sensibilité Δ_{ij} ;
2. Pour k allant de 1 à r,
   (a) calculer la partition $P_{\tau_k} = (S_1, S_2)$;
   (b) calculer les sensibilités associées (Δ_1, Δ_2) ;
   (c) calculer $T_k = F(n_1, n_2, \Delta_1, \Delta_2)$;
3. Retourner l'indice de la valeur minimale dans $T = [T_1, \ldots, T_r]$.

Cet algorithme peut être généralisé pour une partition en $K > 2$ sous-ensembles. En effet, pour des seuils $\tau_1, \ldots, \tau_{K-1}$, la partition $P_{(\tau_i)_1^{K-1}} = (S_1, \ldots, S_K)$ est définie comme suit :

$$\begin{aligned} S_1 &= \{(i,j) \mid \Delta_{ij} \leq \tau_1\}, \\ S_2 &= \{(i,j) \mid \tau_1 < \Delta_{ij} \leq \tau_2\}, \\ &\ldots \\ S_K &= \{(i,j) \mid \tau_{K-1} < \Delta_{ij}\}. \end{aligned}$$

Pour être plus efficace, cette recherche pourrait être effectuée par exemple par dichotomie plutôt qu'avec une recherche exhaustive pour tous les seuils τ_i.

Ajout du bruit selon la sensibilité

Pour chaque sous-ensemble, un bruit calibré à partir de la sensibilité précédente est alors appliqué. Plus les données sont sensibles, plus le bruit ajouté est important. Ainsi, pour chaque coefficient A_{ij} sensible de la matrice A, nous ajoutons un bruit selon la distribution de Laplace dont l'écart-type augmente en fonction de la sensibilité du sous-ensemble S_k auquel appartient A_{ij}. Nous obtenons alors une matrice bruitée $\tilde{A}$ qui satisfait la confidentialité différentielle pour le paramètre ε.

Plus formellement, le mécanisme de Laplace avec une sensibilité par blocs est défini de la manière suivante.

Théorème 11 (Mécanisme de Laplace par blocs)**.** *Soit ε un réel strictement positif qui représente le paramètre de protection de vie privée. Soit $(S_k)_{1 \leq k \leq K}$ une partition de l'ensemble $\mathcal{S}$. Pour k allant de 1 à K, nous définissons les coefficients suivants :*

$$\lambda_k = \frac{\varepsilon_k}{\Delta_k} \text{ avec } \varepsilon_k = \frac{\varepsilon}{\sum_{j=1}^{K} \sqrt{\frac{n_j \Delta_j}{n_k \Delta_k}}}.$$

Le mécanisme de Laplace par blocs $\mathcal{M} : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$ *tel que*

$$\mathcal{M}(A) = A + B = \tilde{A}$$

satisfait la confidentialité différentielle pour ε si les coefficients de la matrice B sont définis par :

– $B_{ij} = 0$, *si* $(i,j) \notin \mathcal{S}$,

– B_{ij} *est une variable aléatoire de Laplace, centrée en 0, d'écart-type* $\sigma_k = \frac{\sqrt{2}}{\lambda_k}$, *si* $(i,j) \in S_k$.

En outre, le choix des coefficients λ_k *est optimal puisqu'ils minimisent l'erreur moyenne* $\varphi : (\varepsilon_1, \ldots, \varepsilon_K) \to \sum_{k=1}^{K} n_k \times \frac{\Delta_k}{\varepsilon_k}$ *avec* $\varepsilon = \sum_{k=1}^{K} \varepsilon_k$.

Démonstration. La preuve du Théorème 11 est disponible à la Section 7.3.2. □

Ainsi, pour une partition $(S_k)_{1 \leq k \leq K}$, l'application du mécanisme de Laplace par blocs est résumée comme suit.

Mécanisme de Laplace par blocs

Entrée : matrice A, paramètre ε et partition $(S_k)_1^K$ de l'ensemble $\mathcal{S}$
Sortie : matrice $\tilde{A}$ satisfaisant la confidentialité différentielle pour ε

1. Pour tout $(i,j) \notin \mathcal{S}$, $\tilde{A}_{ij} = A_{ij}$;
2. Pour tout k allant de 1 à K,
   (a) Calculer la sensibilité $\Delta_k = \max_{A \sim A'} \sum_{(i,j) \in S_k} |A_{ij} - A'_{ij}|$;
   (b) Poser $\lambda_k = \frac{\varepsilon}{\Delta_k} \times \frac{1}{\sum_{j=1}^{K} \sqrt{\frac{n_j \Delta_j}{n_k \Delta_k}}}$;
   (c) Pour tout $(i,j) \in S_k$, $\tilde{A}_{ij} = A_{ij} + Z_k$ avec Z_k, variable aléatoire de Laplace de moyenne 0 et d'écart type $\sigma_k = \frac{\sqrt{2}}{\lambda_k}$;
3. Retourner $\tilde{A}$.

Application de la SVD

Afin de gagner en utilité en réduisant l'impact du bruit, il est possible d'utiliser la technique classique de décomposition en valeurs singulières [BDMN05] en combinaison de notre mécanisme. Pour une matrice C, la décomposition en valeurs singulières permet de déterminer trois matrices U, D et V dans $\mathbb{R}^{n \times n}$ telles que

$$C = U \cdot D \cdot V,$$

avec U et V deux matrices orthogonales et D une matrice diagonale formée des valeurs singulières $\lambda_1 \geq \cdots \geq \lambda_n$.

À partir de D et pour un rang $k \leq n$ fixé, nous obtenons l'approximation C_k de C telle que $C_k = U \cdot D_k \cdot V$ avec D_k une matrice diagonale de rang k qui comprend sur sa diagonale les k plus grandes valeurs singulières de la matrice D tandis que les $n - k$ coefficients restants sont nuls. Cette approximation reprend donc les informations principales contenues dans la matrice C.

Appliquer cette méthode directement à notre matrice bruitée peut entraîner un bruit sur les coefficients non-sensibles. Pour éviter cela, il suffit de ne permettre son application que sur les coefficients sensibles dans $\mathcal{S}$. Pour un mécanisme par blocs, la décomposition en valeurs singulières peut donc être appliquée comme suit.

Mécanisme par blocs et SVD

Entrée : matrice A, paramètre ε (ou ε, δ) et k fixé
Sortie : matrice $\tilde{A}$ satisfaisant la confidentialité différentielle pour ε, de rang k

1. Pour tout $(i,j) \notin \mathcal{S}$, stocker A_{ij};
2. Appliquer le mécanisme par blocs à la matrice A, le résultat obtenu est noté $C = A + B$;
3. Calculer la SVD de la matrice C : $C = UDV$;
4. Calculer son approximation de rang k : $T = UD_kV$;
5. Pour tout $(i,j) \notin \mathcal{S}$, poser $\tilde{A}_{ij} = A_{ij}$
6. Pour tout $(i,j) \in \mathcal{S}$, poser $\tilde{A}_{ij} = T_{ij}$
7. Retourner $\tilde{A}$.

7.2.2 Autres mécanismes possibles

Dans cette section, nous détaillons l'utilisation d'un bruit différent qui est celui de Gauss. D'autres distributions pourraient également être utilisées. Ensuite, nous évoquons le cas où les sensibilités ne sont pas connues du responsable du traitement.

Mécanisme de Gauss par blocs

Dans le cas d'un bruit gaussien, nous considérons une version plus faible de la confidentialité différentielle qui dépend de deux paramètres ε et δ (voir Section 2.3.3). Dans le cas des matrices, un mécanisme $\mathcal{M} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ satisfait la confidentialité différentielle pour (ε, δ) si pour toutes matrices voisines $A \sim A'$ et toute sortie $R \subset \mathbb{R}^{n\times n}$,

$$\mathbb{P}(\mathcal{M}(A) \in R) \leq \exp(\varepsilon) \times \mathbb{P}(\mathcal{M}(A') \in R) + \delta.$$

L'utilisation de variables aléatoires de Gauss est particulièrement bien adaptée à une sensibilité en fonction de la norme L_2. Celle-ci permet alors d'atteindre une meilleure précision car, pour un vecteur donné, la norme L_2 est inférieure à la norme L_1 .

Pour deux matrices voisines A et A', la sensibilité par blocs définie à partir de la norme L_2 discrète est, pour un sous-ensemble $S_k \subset \mathcal{S}$, définie comme suit :

$$\Delta_k^{L_2} = \max_{A\sim A'} \sqrt{\sum_{(i,j)\in S_k} |A_{ij} - A'_{ij}|^2}.$$

Théorème 12 (Mécanisme de Gauss par blocs)**.** *Soit (ε, δ) une paire de réels strictement positifs qui correspondent aux paramètres de protection de la vie privée. Soit $(S_k)_{1\leq k\leq K}$ une partition de l'ensemble $\mathcal{S}$. Pour k allant de 1 à K, nous définissons*

$$\sigma_k = \frac{\Delta_1^{L_2} \times \alpha}{\varepsilon_k} \text{ avec } \varepsilon_k = \frac{\varepsilon\sqrt{\mu_k}}{\sum_{k=1}^{K}\sqrt{\mu_k}}$$

$$\alpha = \sqrt{2\ln\left(\frac{1,25}{\delta_k}\right)} \text{ et } \mu_k = n_k \times \Delta_k^{L_2} \times \alpha$$

tels que $(\delta_k)_k$ *et* ε_k *vérifient les relations suivantes :* $\prod_{k=1}^{K}(1-\delta_k) \geq 1-\delta$ *et* $\sum_{k=1}^{K} \varepsilon_k = \varepsilon$.

Le mécanisme de Gauss par blocs $\mathcal{M} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ *tel que*

$$\mathcal{M}(A) = A + B = \tilde{A}$$

satisfait la confidentialité différentielle pour (ε, δ) *si les coefficients de la matrice B sont :*

- $B_{ij} = 0$*, si* $(i,j) \notin \mathcal{S}$ *;*
- B_{ij} *est une variable aléatoire de Gauss d'écart-type* σ_k*, si* $(i,j) \in S_k$.

En outre, le choix des ε_k *est optimal puisqu'ils minimisent l'erreur moyenne* φ *définie par* $(\varepsilon_1, \ldots, \varepsilon_K) \to \frac{1}{\sqrt{\pi}} \times \sum_{k=1}^{K} \frac{\mu_k}{\varepsilon_k}$ *avec* $\varepsilon = \sum_{k=1}^{K} \varepsilon_k$.

Démonstration. La preuve du Théorème 12 est disponible à la Section 7.3.2. □

Nous avons choisi d'utiliser la norme L_1 pour l'erreur moyenne à minimiser et non pas la norme L_2. De cette façon, nous allons pouvoir comparer ce mécanisme avec celui basé sur Laplace lors de nos expérimentations dans la section suivante.

Ainsi, pour une partition $(S_k)_{1\leq k\leq K}$, l'application du mécanisme de Gauss par blocs est résumée comme suit.

Mécanisme de Gauss par blocs

Entrée : matrice A, paramètres ε et δ, partition $(S_k)_1^K$ de l'ensemble $\mathcal{S}$
Sortie : matrice $\tilde{A}$ satisfaisant la confidentialité différentielle pour (ε, δ)

1. Pour tout $(i,j) \notin \mathcal{S}$, $\tilde{A}_{ij} = A_{ij}$;
2. Pour k allant de 1 à K,
   (a) Calculer la sensibilité $\Delta_k^{L_2} = \max_{A\sim A'} \sum_{(i,j)\in S_k} |A_{ij} - A'_{ij}|^2$;
   (b) Poser $\mu_k = n_k \times \Delta_k^{L_2} \times \sqrt{2\ln\left(\frac{K\times 1{,}25}{2\times\delta_k}\right)}$ et $\varepsilon_k = \frac{\varepsilon\times\sqrt{\mu_k}}{\sum_{j=1}^{K}\sqrt{\mu_k}}$;
   (c) Pour tout $(i,j) \in S_k$, pour tout k, $\tilde{A}_{ij} = A_{ij} + Z_k$ avec Z_k, variable aléatoire de Gauss de moyenne 0 et d'écart type σ_k ;
3. Retourner $\tilde{A}$.

Sensibilité inconnue

Dans certains cas particuliers, le responsable de traitement peut ne pas connaître la sensibilité Δ_{ij} des données à bruiter. Ainsi, dans la situation où les données sont distribuées entre plusieurs entités, chacune d'entre elles n'a accès qu'à une fraction des données à traiter.

Cas linéaire. Nous considérons ici une requête à une dimension $f : \mathcal{D} \to \mathbb{R}$ linéaire telle que :

$$f(D) = \sum_{u\in D} f(D_u)$$

où $u \in D$ signifie que les données D_u de l'individu u sont dans la base de données $D \in \mathcal{D}$.

Pour construire un mécanisme de confidentialité différentielle sans approximation précise de la sensibilité Δ de la fonction f, nous utilisons la technique dite de "troncature de la requête" déjà présente dans l'article [KNRS13b]. Le mécanisme va alors dépendre d'une base de données de référence D_0. La sensibilité D_0 pour f peut être calculée comme suit : $\Delta_{D_0} =$

$\max_{u\in D_0} |f(D_u)|$. L'un des individus qui réalise ce maximum est noté u_0, d'où $\Delta_{D_0} = f(D_{u_0})$. La fonction utilisée est alors f_{D_0}, définie comme suit :

$$f_{D_0}(D_u) = \begin{cases} f(D_u) & \text{si} \quad |f(D_u)| \leq \Delta_{D_0}, \\ f(D_{u_0}) & \text{si} \quad |f(D_u)| > \Delta_{D_0}. \end{cases}$$

Nous définissons alors le mécanisme $\mathcal{M}_{D_0}$ tel que

$$\mathcal{M}_{D_0}(f)(D) = f_{D_0}(D) + Z_{D_0} \text{ pour tout } D \in \mathcal{D}$$

avec Z_{D_0} variable aléatoire de Laplace de paramètre $\frac{\sqrt{2}\Delta_{D_0}}{\varepsilon}$.

Théorème 13. *Le mécanisme $\mathcal{M}_{D_0}$ satisfait la confidentialité différentielle pour le paramètre ε.*

Démonstration. La preuve du Théorème 13 est disponible à la Section 7.3.2. □

Il est important de noter que le responsable de traitement ne peut pas changer la base de données de référence D_0. Par exemple, si D_0 et D_1 sont deux bases distinctes associées aux mécanismes $\mathcal{M}_{D_0}$ et $\mathcal{M}_{D_1}$ qui respectent la confidentialité différentielle pour un paramètre ε, alors la composition de $\mathcal{M}_{D_0}$ et $\mathcal{M}_{D_1}$ ne satisfait pas la confidentialité différentielle pour 2ε contrairement au Théorème 2 de composition séquentielle.

Idéalement, la base D_0 doit refléter le comportement global des bases de données dans $\mathcal{D}$ afin de limiter l'erreur due au passage de f à f_{D_0}. Par exemple, cela est possible en prenant D_0 la plus grande base possible selon la quantité de données à disposition. Ainsi, seulement quelques données aberrantes $D_I \notin D_0$ vont satisfaire l'inégalité $|f(D_I)| > \Delta_{D_0}$.

Enfin, ce mécanisme ne doit pas être confondu avec ceux basés sur des instances de la forme $f(D) + Z_D$ où Z_D dépend de la valeur d'instance D [NRS07].

Cas Laplace par blocs. Dans le cas de notre mécanisme de Laplace par blocs avec Δ_{ij} inconnues, nous considérons que chaque requête φ_{ij} est linéaire et donne un résultat noté A^u_{ij} pour un individu u. La méthode est alors la suivante :

1. choisir la base de données de référence D_0 dans $\mathcal{D}$;
2. calculer la sensibilité de D_0 pour chaque coefficient (i, j) :

$$\Delta_{(D_0,ij)} = \max_{u\in D_0} |A^u_{ij}|;$$

3. exécuter l'algorithme de recherche de partition avec $\Delta_{(D_0,ij)}$ pour obtenir $(S_{(D_0,1)}, S_{(D_0,2)})$ et les sensibilités correspondantes $(\Delta_{(D_0,1)}, \Delta_{(D_0,2)})$;
4. pour chaque $1 \leq k < K$, trouver u^k_0 qui réalise le maximum $\Delta_{(D_0,k)} = \sum_{(i,j)\in S_{(D_0,k)}} |A^{u_0}_{ij}|$;
5. définir la troncature A_{D_0} de la matrice A telle que pour tout $(i, j) \in S_{(D_0,k)}$,

$$A^u_{(D_0,ij)} = \begin{cases} A^u_{ij} & \text{si} \sum_{(i,j)\in S_{(D_0,k)}} |A^u_{ij}| \leq \Delta_{(D_0,k)}, \\ A^{u^k_0}_{ij} & \text{sinon ;} \end{cases}$$

6. pour ε, définir le mécanisme $\mathcal{M}_{D_0} : \mathbb{R} \to \mathbb{R}$ tel que

$$\mathcal{M}_{D_0}(A) = A_{D_0} + B = \tilde{A}$$

où B est une matrice dont chaque coefficient B_{ij} vaut 0 s'il est non-sensible et est une variable aléatoire sinon, comme précédemment.

Théorème 14. *Pour φ_{ij} linéaire, le mécanisme $\mathcal{M}_{D_0}$ de Laplace par blocs pour des sensibilités non connues satisfait la confidentialité différentielle.*

Démonstration. La preuve du Théorème 14 est disponible à la Section 7.3.2. □

7.3 Évaluation expérimentale et preuves des théorèmes

Dans cette section, nous comparons nos mécanismes par blocs aux mécanismes existants à partir d'une expérimentation sur des données de compte-rendus d'appels. Nous réalisons ensuite les preuves des différents théorèmes.

7.3.1 Expérimentation

Nous allons ici étudier les différents mécanismes : ceux de Laplace et Gauss classiques, notés L et G ; leur version par blocs, notées BL et BG ; une approximation standard de rang k avec la décomposition en valeurs singulières SVD ; les mécanismes L et G avec la SVD, notés alors LSVD et GSVD et enfin leur version par blocs avec la SVD notées BLSVD et BGSVD. Pour comparer ces mécanismes, nous utilisons un même niveau de vie privée donnée par ε et δ.

Cadre de l'expérimentation

Méthode pour l'évaluation des résultats. Dans la littérature, la distance classique pour comparer deux matrices est la norme de Frobenius, induite par la norme L_2 sur les coefficients. Pour évaluer nos résultats, nous utilisons une légère variante de cette distance en fonction de la norme L_1. Ainsi, pour deux matrices A et B dans $\mathbb{R}^{n\times n}$, nous utilisons la formule suivante :

$$|A-B|_1 = \sum_{i,j\in\{1,n\}} |A_{ij} - B_{ij}|.$$

Pour les algorithmes utilisant un bruit aléatoire de Laplace, cette mesure est plus naturelle car elle est plus proche de la définition de sensibilité utilisée. La norme L_2 sera quant-à-elle privilégiée dans le cas de bruit gaussien.

En outre, étant donné que nos bases de données et matrices sont particulièrement grandes, il n'est pas évident d'en évaluer la qualité. Pour faciliter cela, nous normalisons la distance précédente. Nous choisissons aléatoirement $2l$ bases de données, notées $D_1, \ldots, D_{2l}$ dans $\mathcal{D}$, de taille N et considérons donc les matrices $A_1 = \psi(D_1), \ldots, A_{2l} = \psi(D_{2l})$. Nous définissons le facteur de normalisation d_l :

$$d_l = \frac{1}{l} \sum_{i=1}^{l} |A_{2i-1} - A_{2i}|_1.$$

Pour évaluer deux matrices A et B, la distance d est alors définie comme suit :

$$d(A,B) = \frac{1}{d_l}|A-B|_1.$$

Ainsi, pour de grandes bases de données, obtenir $d(A,\tilde{A})$ proche de 1 est considéré comme un bon résultat. Autrement dit, $\tilde{A}$ garde bien les informations statistiques de la matrice initiale A.

Base de données (CRA). Nous considérons ici une base de données de comptes-rendus d'appels d'un opérateur téléphonique. Cette base de données contient les informations suivantes : date, heure et localisation de l'antenne la plus proche lors d'appels et de SMS reçus ou émis. Il est donc possible de construire des matrices de mobilité des utilisateurs, comme expliqué dans notre exemple de mobilité au cours de la description de notre mécanisme. Nous comptons les transitions entre les antennes sur une période de deux semaines. Les groupes considérés sont de $N = 33000$ utilisateurs et l'implémentation a été réalisée en Scala avec la librairie Big Data de Spark [ZXW+16].

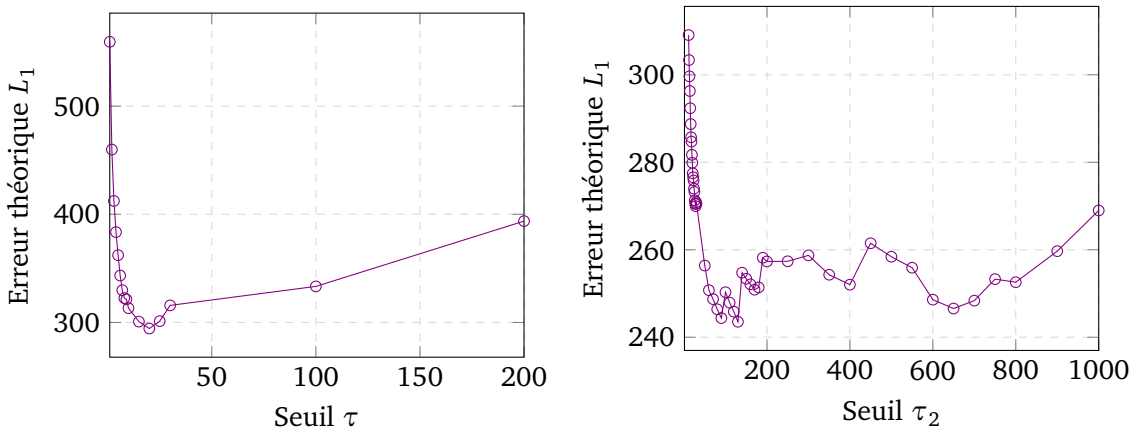
Étude des résultats

Choix de la partition. Pour choisir la partition, il est possible d'utiliser l'algorithme de recherche présenté précédemment. Pour un nombre de partitions K et une fonction d'erreur F, l'algorithme calcule une approximation du meilleur seuil τ pour K égal à 2 ou des différents $(\tau_i)_i$ si K est supérieur à 2. Dans notre cas, nous avons choisi F pour minimiser l'erreur moyenne L_1. La Figure 7.2 montre la dépendance entre l'erreur théorique du mécanisme de Laplace par blocs BL et le(s) seuil(s) choisi(s).

En particulier, la Figure 7.2a illustre le cas d'une partition en deux sous-ensembles. Dans le cas d'un mécanisme de Laplace par blocs, les valeurs optimales pour τ se situent entre 10 et 20 tandis que, dans le cas de Gauss par blocs, le seuil idéal est environ 35. Pour une partition en trois sous-ensembles, la Figure 7.2b montre la recherche du deuxième seuil pour τ_1 fixé à 10. La courbe obtenue est plus complexe que la précédente car plus de paramètres impactent le choix de ce seuil à savoir le nombre d'éléments et la sensibilité de chaque sous-ensemble de la partition.

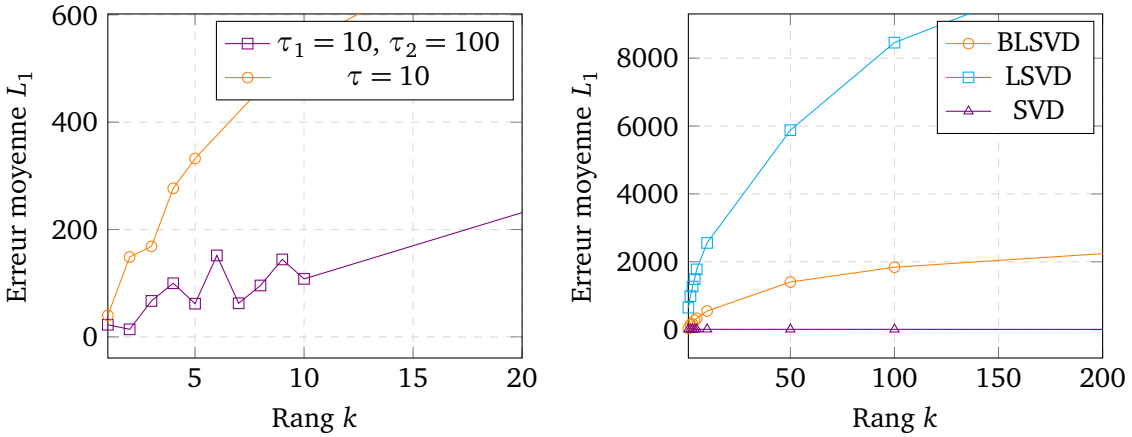
Cas d'un bruit laplacien. Pour 33 000 utilisateurs, nous notons que la suite $(d_l)_l$ converge lorsque l atteint 10. La Figure 7.3a montre qu'un petit paramètre K, qui définit la taille de la partition, est suffisant pour que notre mécanisme de Laplace par blocs BL soit plus efficace que le mécanisme de Laplace classique. Bien sûr, nous supposons que pour des applications différentes avec des structures de graphes plus complexes, l'utilisation de plus de sous-ensembles peut permettre d'avoir de meilleures performances.

En outre, la Figure 7.3b montre que les mécanismes basés sur la sensibilité par blocs sont meilleurs que ceux avec une sensibilité globale. Cependant, il est important de noter que les résultats obtenus avec le mécanisme de Laplace par blocs BL restent loin d'être raisonnables puisque, dans le meilleur des cas, nous obtenons une distance $d(A,\tilde{A})$ à environ 2000. En revanche, les mécanismes avec SVD, à savoir BLSVD et LSVD atteignent des valeurs acceptables avec une distance $d(A,\tilde{A})$ proche de 1, même pour un niveau de respect de la vie privée assez



(a) Recherche τ pour $K = 2$ (b) Recherche τ_2 pour $K = 3$ et $\tau_1 = 10$

FIGURE 7.2 – Recherche du meilleur seuil pour $\varepsilon = 1$



(a) BLSVD pour deux partitions $K = 2$ ou $K = 3$ (b) BLSVD, LSVD et SVD pour $K = 2$ et $\tau = 10$

FIGURE 7.3 – Comparaison des différents mécanismes laplaciens pour $\varepsilon = 0,1$.

haut avec ε égal à 0,1. De plus, pour K suffisamment petit, l'erreur est proche de celle d'une approximation de rang k simple, sans ajout de bruit. Autrement dit, nous avons $d(A,\tilde{A}) \sim d(A,A_k)$ avec A_k matrice résultant de la SVD. Cela signifie que les plus petites valeurs propres et respectivement vecteurs propres de la matrice assainie sont proches, au sens de la distance d, de celles d'une matrice non bruitée. Ces résultats étaient déjà présents dans [MM09] pour le mécanisme LSVD. Nos travaux permettent de montrer que pour toute valeur de K, le mécanisme BLSVD permet une anonymisation de meilleure qualité qu'avec LSVD. La prise en compte de la sensibilité par blocs peut donc permettre d'améliorer les résultats antérieurs.

Cas d'un bruit gaussien. Pour comparer nos mécanismes par blocs basés sur la distribution de Gauss à ceux basés sur celle de Laplace, nous utilisons la norme L_1 pour mesurer les erreurs. La Figure 7.4 reprend les mécanismes BL et BG ainsi que leurs versions classiques avec une sensibilité globale. Le taux d'erreur est bien inférieur dans le cas de la sensibilité par blocs.

Enfin, contrairement aux mécanismes par blocs avec Gauss ou Laplace, la dépendance par

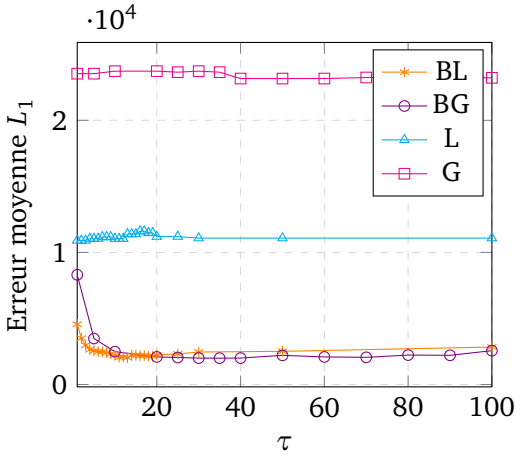
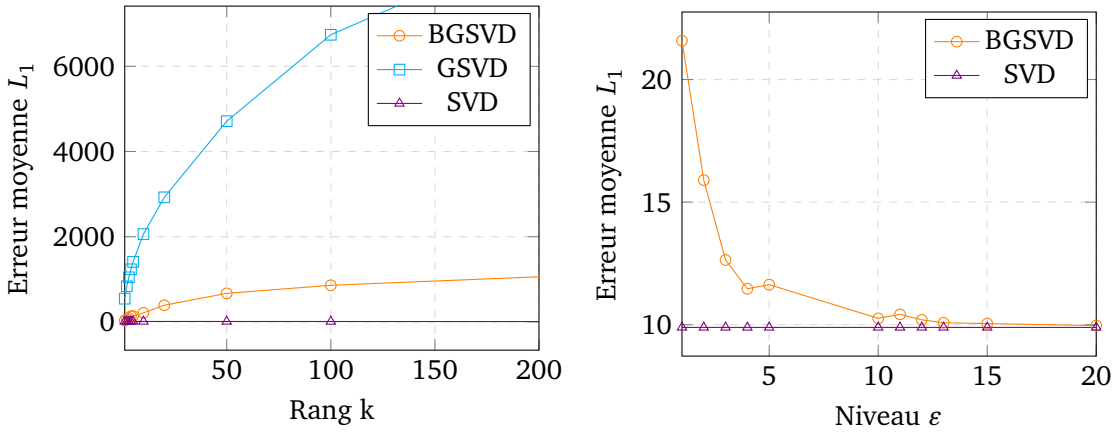


FIGURE 7.4 – Mécanismes BL, L, BG et G pour $\varepsilon = 0,1$, $\delta = 0,001$ et $K = 2$



(a) BLSVD, GSVD et SVD pour $\varepsilon = 0,1$

(b) BGSVD, SVD pour rang $k = 5$

FIGURE 7.5 – Comparaison des mécanismes gaussiens pour $\delta = 0,001$, $K = 2$ et $\tau = 35$.

rapport à la valeur ε n'est pas linéaire si la SVD est appliquée avec BGSVD et BLSVD. La Figure 7.5 regroupe deux courbes qui montrent comme le mécanisme BGSVD et la SVD, donc sans perturbation, sont proches pour un même rang k. Ainsi, les données telles que vecteurs et valeurs propres sont préservées en utilisant le mécanisme de Gauss par blocs suivi de la SVD avec le paramètre δ à 0,001 et dès que ε supérieur ou égal à 3.

7.3.2 Preuves

Preuve du Théorème 11

Paramètres (ε_k). Soit $\varepsilon_k = \varepsilon \times \frac{\sqrt{n_k \Delta_k}}{\sum_{j=1}^{K} \sqrt{n_j \Delta_j}}$ tel que $\lambda_k = \frac{\varepsilon_k}{\Delta_k}$. Il est immédiat que $\sum_{k=1}^{K} \varepsilon_k = \varepsilon$.

Mécanisme $\mathcal{M}$. Par définition, le mécanisme $\mathcal{M}$ satisfait la confidentialité différentielle pour le paramètre ε si et seulement si, pour toute matrice $A' \sim A$ et tout $S = (S_{ij})_{ij} \in \mathbb{R}^{n \times n}$, nous avons :

$$\frac{\mathbb{P}(\tilde{A} \in S)}{\mathbb{P}(\tilde{A}' \in S)} \leq \exp(\varepsilon).$$

Étant donné que les $\tilde{A}_{ij}$ sont des variables aléatoires indépendantes et que $\tilde{A}_{ij} = A_{ij} = A'_{ij}$ pour $(i,j) \notin \mathcal{S}$, nous devons montrer que, pour tout $S = (S_{ij})_{ij} \in \mathbb{R}^{n \times n}$:

$$\prod_{(i,j)\in\mathcal{S}} \frac{\mathbb{P}(\tilde{A}_{ij} \in S_{ij})}{\mathbb{P}(\tilde{A}'_{ij} \in S_{ij})} \leq \exp(\varepsilon).$$

Or, nous avons $g_{\tilde{A}_{ij}}(y) = \mu \times \exp(-\lambda_{ij}|y - A_{ij}|)$ et $g_{\tilde{A}'_{ij}}(y) = \mu \times \exp(-\lambda_{ij}|y - A'_{ij}|)$ pour μ coefficient de normalisation. La condition précédente est équivalente, pour y_{ij} dans $\mathbb{R}^{n \times n}$, à :

$$\prod_{(i,j)\in\mathcal{S}} \frac{g_{\tilde{A}_{ij}}(y_{ij})}{g_{\tilde{A}'_{ij}}(y_{ij})} \leq \exp(\varepsilon).$$

À partir de l'inégalité triangulaire, nous avons :

$$\begin{aligned} \frac{g_{\tilde{A}_{ij}}(y_{ij})}{g_{\tilde{A}'_{ij}}(y_{ij})} &= \exp\Big(-\lambda_{ij} \times (|y_{ij} - A_{ij}| - |y_{ij} - A'_{ij}|)\Big) \\ &\leq \exp\Big(-\lambda_{ij} \times |A_{ij} - A'_{ij}|\Big). \end{aligned}$$

Ainsi, par la définition de la sensibilité, nous avons :

$$\begin{aligned} \prod_{(i,j)\in\mathcal{S}} \frac{g_{\tilde{A}_{ij}}(y_{ij})}{g_{\tilde{A}'_{ij}}(y_{ij})} &\leq \prod_{(i,j)\in\mathcal{S}} \exp\Big(\lambda_{ij} \times |A_{ij} - A'_{ij}|\Big) \\ &= \exp\left(\sum_{(i,j)\in\mathcal{S}} \lambda_{ij}|A_{ij} - A'_{ij}|\right) \\ &= \exp\left(\sum_{k=1}^{K} \lambda_k \times \sum_{(i,j)\in S_k} |A_{ij} - A'_{ij}|\right) \\ &\leq \exp\left(\sum_{k=1}^{K} \lambda_k \times \Delta_k\right) \\ &= \exp\left(\sum_{k=1}^{K} \varepsilon_k\right) \\ &\leq \exp(\varepsilon). \end{aligned}$$

Le mécanisme $\mathcal{M}$ satisfait donc la confidentialité différentielle pour ε.

Coefficients (λ_k). Nous devons prouver que notre choix des $(\lambda_k)_k$ et donc des $(\varepsilon_k)_k$ minimise l'erreur moyenne L_1 sur ces coefficients. Cette erreur ERR est définie par : ERR =

$\mathbb{E}\left(\sum_{ij} |A_{ij} - \tilde{A}_{ij}|\right)$. Nous avons alors les relations suivantes :

$$\begin{aligned} \text{ERR} &= \sum_{(i,j)\in\mathcal{S}} \mathbb{E}(|B_{ij}|) \\ &= \sum_{k=1}^{K} \sum_{(i,j)\in S_k} \mathbb{E}(|B_{ij}|) \\ &= \sum_{k=1}^{K} n_k \times \frac{\lambda_k}{\varepsilon_k}. \end{aligned}$$

Il est alors évident que le choix des $(\varepsilon_k)_k$ tels que $\varepsilon_k = \varepsilon \times \frac{\sqrt{n_k \Delta_k}}{\sum_{j=1}^{K} \sqrt{n_j \Delta_j}}$ minimise la fonction $\varphi(\varepsilon_1, \ldots, \varepsilon_K) = \sum_{k=1}^{K} n_k \times \frac{\lambda_k}{\varepsilon_k}$ sous la contrainte $\bar{\varphi}(\varepsilon_1, \ldots, \varepsilon_K) = \varepsilon = \sum_{k=1}^{K} \varepsilon_k$.

En utilisant les multiplicateurs de Lagrange, un extremum local pour φ satisfait, pour un scalaire μ, $\text{grad}\varphi(\varepsilon_1, \ldots, \varepsilon_K) = \mu \times \text{grad}\bar{\varphi}(\varepsilon_1, \ldots, \varepsilon_K)$. Avec la contrainte précédente, cette équation nous donne bien la forme des $(\varepsilon_k)_k$ comme définie dans le théorème. En particulier, un tel extremum est unique et est évidemment un minimum, ce qui conclut la preuve.

Preuve du Théorème 12

Pour démontrer que le mécanisme $\mathcal{M}$ satisfait la confidentialité différentielle pour (ε, δ), il suffit de montrer que pour toute matrice A dans $\mathbb{R}^{n\times n}$, il existe un sous-ensemble E_A inclus dans $\mathbb{R}^{n\times n}$ tel que :

1. $\frac{g_{\tilde{A}}(Y)}{g_{\tilde{A}'}(Y)} \leq \exp(\varepsilon)$ pour tout Y dans E_A avec $A' \sim A$;
2. $\mathbb{P}(\tilde{A} \notin E_A) \leq \delta$.

Pour prouver ces conditions, nous posons $E_A = \bigcap_{1\leq k<K} E_{A,k}$ tel que

$$E_{A,k} = \left\{ Y = (Y_{ij})_{ij} \in \mathbb{R}^{n\times n} \mid \lambda_k (\Delta_k^{L_2})^2 + \left(2\lambda_k \Delta_k^{L_1} \times \max_{(i,j)\in S_k} |A_{ij} - Y_{ij}| \right) \leq \varepsilon_k \right\}.$$

Condition 1. Le terme $\frac{g_{\tilde{A}}(Y)}{g_{\tilde{A}'}(Y)}$ peut être réécrit à partir de :

$$\begin{aligned} g_{\tilde{A}}(Y) &= \prod_{(i,j)\in\mathcal{S}} g_{\tilde{A}_{ij}}(Y_{ij}) \\ &= \prod_{k=1}^{K} \prod_{(i,j)\in S_k} \exp(-\lambda_k |A_{ij} - Y_{ij}|^2) \\ &= \exp\left(\sum_{k=1}^{K} \sum_{(i,j)\in S_k} -\lambda_k |A_{ij} - Y_{ij}|^2 \right). \end{aligned}$$

La condition 1 est alors vérifiée puisque que $\sum_{k=1}^{K} \varepsilon_k = \varepsilon$ et que nous avons les inégalités

suivantes, pour toute matrice $A \sim A'$ et quelque soit k de 1 à K :

$$\begin{aligned}\sum_{(i,j)\in S_k} \lambda_k \times |(A_{ij}-Y_{ij})^2-(A'_{ij}-Y_{ij})^2| &\leq \sum_{(i,j)\in S_k} \lambda_k \times (|A_{ij}-A'_{ij}|^2 + 2\times|A_{ij}-A'_{ij}|\times|A_{ij}-Y_{ij}|)\\ &\leq \lambda_k(\Delta_k^{L_2})^2 + (2\lambda_k\Delta_k^{L_1} \times \max_{(i,j)\in\mathcal{S}} |A_{ij}-Y_{ij}|) \leq \varepsilon_k.\end{aligned}$$

Condition 2. Étant donné le Théorème A.1. de l'article [DR14], nous avons

$$\mathbb{P}(\tilde{A}\in E_{A,k}) \geq 1-\delta_k \text{ pour tout } 1\leq k\leq K.$$

Or, $\prod_{k=1}^{K}(1-\delta_k) \geq 1-\delta$ par hypothèse et nous avons l'inégalité suivante :

$$\begin{aligned}\mathbb{P}(\tilde{A}\in E_A) &= 1-\mathbb{P}(\tilde{A}\in E_A)\\ &= 1-\prod_{k=1}^{K}\mathbb{P}(\tilde{A}\in E_{A,k})\\ &\leq 1-\prod_{k=1}^{K}(1-\delta_k)\\ &\leq \delta.\end{aligned}$$

Ainsi, le mécanisme $\mathcal{M}$ satisfait la confidentialité différentielle pour les paramètres ε et δ. Concernant les paramètres ε_k qui minimisent l'erreur moyenne L_1, la preuve suit celle du Théorème 11 en utilisant $\mathbb{E}(|Z|) = \frac{1}{\sqrt{\pi}\times\sqrt{\lambda}}$ pour une variable aléatoire de Gauss d'écart-type $\sigma = \frac{1}{\sqrt{\lambda}}$. Pour les paramètres δ_k, il est possible de choisir un même δ_0 identique pour tout k de 1 à K. En effet, nous avons $\delta_k \sim \frac{\delta}{K}$ lorque δ est petit et que K l'est également suffisamment. Par conséquent, pour un paramètre $\delta > 0$ suffisamment petit, le choix de $\delta_k = \frac{2\times\delta}{K}$ pour tout k permet de satisfaire la confidentialité différentielle (ε, δ).

Preuve du Théorème 13

Étant donné que $|f_{D_0}(D)-f_{D_0}(D')| \leq \Delta_{D_0}$ pour toutes bases voisines D et D' dans $\mathcal{D}$, nous avons les inégalités suivantes pour les deux cas possibles $D \sim D'$:

1. $|f(D)-f(D')| = |f(D_I)| \leq \Delta_{D_0}$ d'où $f_{D_0}(D)-f_{D_0}(D') = f_{D_0}(D_I) = f(D_I)$;
2. $|f(D)-f(D')| = |f(D_I)| \leq \Delta_{D_0}$ d'où $f_{D_0}(D)-f_{D_0}(D') = f_{D_0}(D_I) = f(D_{I_0})$.

La preuve suit alors le même déroulement que celle du Théorème 11.

Preuve du Théorème 14

Cette preuve découle de celles des deux Théorèmes précédents 11 et 13. Nous considérons deux matrices voisines $A \sim \tilde{A}$ et des coefficients (i,j) dans $S_{(D_0,k)}$.

Par linéarité des requêtes φ_{ij}, nous avons $|A_{(D_0,ij)}-A'_{(D_0,ij)}| = |A^I_{(D_0,ij)}|$ pour un individu I. Par définition de la troncature $A \mapsto A_{D_0}$, $|A^I_{(D_0,ij)}| \leq \Delta_{(D_0,k)}$. Nous posons

$$\varepsilon_{(D_0,k)} = \frac{\varepsilon\sqrt{n_{(D_0,k)}\Delta_{(D_0,k)}}}{\sum_{j=1}^{K}\sqrt{n_{(D_0,j)}\Delta_{(D_0,j)}}}$$

tel que $\lambda_{(D_0,k)} = \frac{\varepsilon_{(D_0,k)}}{\Delta_{(D_0,k)}}$. Nous avons alors :

$$\begin{aligned}
\prod_{(i,j)\in\mathcal{S}} \frac{g_{\tilde{A}_{ij}}(y_{ij})}{g_{\tilde{A}'_{(D_0,ij)}}(y_{(D_0,ij)})} &\leq \prod_{(i,j)\in\mathcal{S}} \exp\left(\lambda_{(D_0,ij)} \times |A_{(D_0,ij)} - A'_{(D_0,ij)}|\right) \\
&= \exp\left(\sum_{(i,j)\in\mathcal{S}} \lambda_{(D_0,ij)} |A_{(D_0,ij)} - A'_{(D_0,ij)}|\right) \\
&= \exp\left(\sum_{k=1}^{K} \lambda_{(D_0,k)} \times \sum_{(i,j)\in S_{(D_0,k)}} |A_{(D_0,ij)} - A'_{(D_0,ij)}|\right) \\
&\leq \exp\left(\sum_{k=1}^{K} \lambda_{(D_0,k)} \times \Delta_{(D_0,k)}\right) \\
&= \exp\left(\sum_{k=1}^{K} \varepsilon_{(D_0,k)}\right) \\
&\leq \exp(\varepsilon).
\end{aligned}$$

Le mécanisme $\mathcal{M}_{D_0}$ satisfait donc la confidentialité différentielle pour ε.

Conclusion

Dans ce chapitre, nous avons étudié comment anonymiser des graphes pour respecter la confidentialité différentielle. Notre nouveau mécanisme permet de préserver une bonne utilité des données. Pour cela et contrairement à la quasi-totalité des systèmes proposés dans l'état de l'art, le bruit est appliqué différemment à plusieurs sous-ensembles selon leur propre sensibilité. Les résultats obtenus montrent que l'utilité finale est meilleure qu'avec un bruit uniforme. En outre, ce schéma ne fait pas d'hypothèse particulière sur le graphe considéré et peut donc s'adapter à de nombreuses situations.

Conclusion

Dans ce mémoire, nous avons étudié comment concilier le respect de la vie privée avec l'utilité et la performance des services proposés. Pour cela, nous avons suivi deux axes distincts.

Tout d'abord, nous avons introduit trois nouvelles primitives : un MAC algébrique, un MAC séquentiellement agrégé ainsi qu'une signature partiellement aveugle. Chacune de ces constructions représente une contribution à part entière et a été prouvée sûre. Nous les avons ensuite utilisées pour concevoir trois schémas qui bénéficient alors de leur efficacité.

Nous nous sommes intéressés aux accréditations anonymes avec vérification par clé. Cette variante à clé secrète des accréditations anonymes classiques, introduites par CHAUM, permet d'obtenir de meilleures performances. Grâce à notre MAC algébrique, nous avons proposé un schéma où une unique clé suffit quelque soit le nombre d'attributs certifiés, contrairement aux systèmes existants. Une simple modification côté vérifieur permet d'adapter ce système pour des vérifications publiques. Il est non-chaînable, quelque soit le nombre d'utilisations, tout en étant aussi efficace que le schéma *U-Prove* de Microsoft qui n'assure pas cette propriété.

Ensuite, nous avons considéré le cas particulier du vote électronique qui soulève de nombreuses problématiques. Nous nous sommes particulièrement intéressés à la propriété de résistance à la coercition et à la possibilité de réaliser plusieurs élections successives à partir d'une même accréditation. Nous avons alors décrit un nouveau schéma de vote électronique. Celui-ci est résistant à la coercition et repose sur des accréditations qui peuvent être mises à jour pour un nouveau scrutin, grâce à notre MAC séquentiellement agrégé. Il offre ainsi la possibilité de révoquer certains électeurs si besoin. Sa complexité linéaire le rend utilisable pour de réelles élections.

De la même façon, nous avons traité le paiement électronique anonyme. Nous avons ciblé des cas d'usages spécifiques où une même entité joue à la fois le rôle de la banque et du marchand. Les schémas existants ne parvenaient pas à concilier à la fois les performances de sécurité et l'efficacité pour des périphériques aux capacités limitées. Pour ces applications, comme le télépéage, nous avons alors décrit un nouveau schéma de paiement électronique dit privé qui assure la non-chaînabilité des transactions, avec notre signature partiellement aveugle. Son implémentation sur carte SIM a montré qu'un paiement pouvait être réalisé en seulement 205 ms. Cela prouve son efficacité et sa possible utilisation en pratique.

Dans une dernière partie, nous avons étudié le cas de l'anonymisation pour des données stockées qui peuvent se représenter sous forme de graphes. Nous nous sommes concentrés en particulier sur la notion de confidentialité différentielle. Pour limiter la quantité de bruit ajouté afin d'assurer le respect de la vie privée, nous avons introduit un nouveau mécanisme de bruitage dit par blocs. Le bruit est appliqué en fonction d'une sensibilité qui considère des sous-ensembles de données au lieu de la base entière en une seule fois. Comme le montre notre

expérimentation, ce mécanisme permet d'obtenir des données moins altérées et qui respectent toujours la définition de la confidentialité différentielle.

Perspectives

Nos travaux montrent qu'il est aujourd'hui possible d'obtenir des services sécurisés, efficaces et qui garantissent le respect de la vie privée. Néanmoins, ces constructions concernent des cas d'usages très spécifiques.

D'une part, chaque application implique ses propres besoins en terme de sécurité et de vie privée. Les solutions proposées ne sont pas toutes universelles et doivent être adaptées. Ainsi, l'utilisation de compteurs communicants, de capteurs d'activités ou encore de véhicules connectés sont autant de cas d'usages qui collectent aujourd'hui de nombreuses données personnelles. Il serait intéressant d'étudier si nos constructions peuvent ici permettre d'assurer le respect de la vie privée de l'utilisateur vis-à-vis du fournisseur de service.

D'autre part, l'anonymisation est un sujet particulièrement actuel. L'analyse des données ou des comportements d'utilisateurs est aujourd'hui au cœur des services que nous utilisons. L'anonymisation permet de conserver l'information contenue dans les données tout en préservant la vie privée des individus. Néanmoins, une solution parfaite est impossible. Elle entraîne obligatoirement une part de risque qu'il n'est pas évident de quantifier. Par exemple, il n'existe pas de valeur sûre pour le choix du paramètre ε sur lequel repose la confidentialité différentielle. Ainsi, il reste de nombreux travaux à mener sur l'application de mécanismes d'anonymisation selon le type de données traitées et la finalité recherchée. De même, la compréhension de ces mécanismes et l'impact du choix des paramètres n'est pas facile à appréhender pour l'utilisateur. Concevoir un modèle unifié et compréhensible par les non-initiés serait une réelle avancée.

Enfin, il est important de mentionner que les utilisateurs prennent désormais de plus en plus conscience des problématiques de protection de leurs données personnelles. Le respect de la vie privée tout comme la sécurité ne doivent pas être vus comme des contraintes techniques. Il est essentiel de chercher de nouvelles solutions adaptées et d'anticiper les futurs usages.

Publications et brevet

Publications Internationales

[BBD+17] *Private eCash in Practice (Short Paper)*
Financial Cryptography and Data Security 2016
A. Barki, S. Brunet, N. Desmoulins, S. Gambs, S. Gharout et J. Traoré

[ABBT16] *Remote Electronic Voting can be Efficient, Verifiable and Coercion-Resistant*
VOTING 2016
R. Araùjo, A. Barki, S. Brunet et J. Traoré

[BBDT16] *Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials*
Selected Areas in Cryptography 2016
A. Barki, S. Brunet, N. Desmoulins et J. Traoré

[BCGO16] *Edge-Calibrated Noise for Differentially Private Mechanisms on Graphs*
Privacy, Security and Trust 2016
S. Brunet, S. Canard, S. Gambs et B. Olivier

Brevet

Procédé et dispositif d'anonymisation de données stockées dans une base de données
Déposé le 09/02/2016, FR1651021
S. Brunet, S. Canard, D. Le Hello et B. Olivier

Bibliographie

[ABBT16] Roberto ARAÚJO, Amira BARKI, Solenn BRUNET, and Jacques TRAORÉ. Remote electronic voting can be efficient, verifiable and coercion-resistant. In Jeremy CLARK, Sarah MEIKLEJOHN, Peter Y. A. RYAN, Dan S. WALLACH, Michael BRENNER, and Kurt ROHLOFF, editors, *FC 2016 Workshops*, volume 9604 of *LNCS*, pages 224–232. Springer, Heidelberg, February 2016. doi:10.1007/978-3-662-53357-4_15.

[Abe01] Masayuki ABE. A secure three-move blind signature scheme for polynomially many signatures. In Birgit PFITZMANN, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Heidelberg, May 2001.

[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. URL : http://dx.doi.org/10.1109/SFCS.1979.2, doi:10.1109/SFCS.1979.2.

[AF96] Masayuki ABE and Eiichiro FUJISAKI. How to date blind signatures. In Kwangjo KIM and Tsutomu MATSUMOTO, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 244–251. Springer, Heidelberg, November 1996.

[AFT08] Roberto ARAÚJO, Sébastien FOULLE, and Jacques TRAORÉ. A practical and secure coercion-resistant scheme for remote elections. In David CHAUM, Miroslaw KUTYLOWSKI, Ronald L. RIVEST, and Peter Y. A. RYAN, editors, *Frontiers of Electronic Voting*, number 07311 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2008. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[AJL13] Faraz AHMED, Rong JIN, and Alex X. LIU. A random matrix approach to differential privacy and structure preserved social network graph publishing. *CoRR*, abs/1307.0475, 2013.

[ALF+14] Man Ho AU, J.K. LIU, Junbin FANG, Z.L. JIANG, W. SUSILO, and Jianying ZHOU. A new payment system for enhancing location privacy of electric vehicles. *IEEE Transactions on Vehicular Technology*, 63(1) :3–18, Jan 2014. doi:10.1109/TVT.2013.2274288.

[ALT+15] Ghada ARFAOUI, Jean-François LALANDE, Jacques TRAORÉ, Nicolas DESMOULINS, Pascal BERTHOMÉ, and Saïd GHAROUT. A practical set-membership proof for privacy-preserving NFC mobile ticketing. *Proceedings on Privacy Enhancing Technologies*, abs/1505.03048, 2015.

[AMO08] Norio AKAGI, Yoshifumi MANABE, and Tatsuaki OKAMOTO. An efficient anonymous credential system. In Gene TSUDIK, editor, *FC 2008*, volume 5143 of *LNCS*, pages 272–286. Springer, Heidelberg, January 2008.

[ANS13] Référentiel général de sécurité. Technical report, Agence Nationale de la Sécurité des Systèmes d'Information, 2013.

[AT13] Roberto ARAÚJO and Jacques TRAORÉ. A practical coercion resistant voting scheme revisited. In HEATHER, *J.,* SCHNEIDER, *S.,* TEAGUE, *V. (eds.) Vote-ID 2013*, volume 7985 of *LNCS*, pages 193–209. Springer, Heidelberg, 2013.

[BB04] Dan BONEH and Xavier BOYEN. Short signatures without random oracles. In Christian CACHIN and Jan CAMENISCH, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004.

[BBD+17] Amira BARKI, Solenn BRUNET, Nicolas DESMOULINS, Sébastien GAMBS, Saïd GHAROUT, and Jacques TRAORÉ. *Private eCash in Practice (Short Paper)*, pages 99–109. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017. doi:10.1007/978-3-662-54970-4_6.

[BBDT16] Amira BARKI, Solenn BRUNET, Nicolas DESMOULINS, and Jacques TRAORÉ. *Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials*. 2016.

[BBS04] Dan BONEH, Xavier BOYEN, and Hovav SHACHAM. Short group signatures. In Matthew FRANKLIN, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

[BCC88] Gilles BRASSARD, David CHAUM, and Claude CRÉPEAU. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2) :156–189, 1988.

[BCGO16] S. BRUNET, S. CANARD, S. GAMBS, and B. OLIVIER. Edge-calibrated noise for differentially private mechanisms on graphs. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 42–49, Dec 2016. doi:10.1109/PST.2016.7906935.

[BDMN05] Avrim BLUM, Cynthia DWORK, Frank MCSHERRY, and Kobbi NISSIM. Practical privacy : The sulq framework. In *Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, pages 128–138, New York, NY, USA, 2005. ACM.

[BDPR98] Mihir BELLARE, Anand DESAI, David POINTCHEVAL, and Phillip ROGAWAY. Relations among notions of security for public-key encryption schemes. In Hugo KRAWCZYK, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer, Heidelberg, August 1998.

[BG90] Mihir BELLARE and Shafi GOLDWASSER. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles BRASSARD, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, Heidelberg, August 1990.

[BGLS03] Dan BONEH, Craig GENTRY, Ben LYNN, and Hovav SHACHAM. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli BIHAM, editor, *EURO-*

CRYPT 2003, volume 2656 of *LNCS*, pages 416–432. Springer, Heidelberg, May 2003.

[BGN05] Dan BONEH, Eu-Jin GOH, and Kobbi NISSIM. Evaluating 2-DNF formulas on ciphertexts. In Joe KILIAN, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, Heidelberg, February 2005.

[BL13] Foteini BALDIMTSI and Anna LYSYANSKAYA. Anonymous credentials light. In Ahmad-Reza SADEGHI, Virgil D. GLIGOR, and Moti YUNG, editors, *ACM CCS 13*, pages 1087–1098. ACM Press, November 2013.

[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*, volume 3897 of *LNCS*, pages 319–331. Springer, Heidelberg, August 2006.

[BNPS03] Mihir BELLARE, Chanathip NAMPREMPRE, David POINTCHEVAL, and Michael SEMANKO. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3) :185–215, June 2003.

[BP97] Niko BARI and Birgit PFITZMANN. Collision-free accumulators and fail-stop signature schemes without trees. In Walter FUMY, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 480–494. Springer, Heidelberg, May 1997.

[BR93] Mihir BELLARE and Phillip ROGAWAY. Random oracles are practical : A paradigm for designing efficient protocols. In V. ASHBY, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BR04] Mihir BELLARE and Phillip ROGAWAY. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. `http://eprint.iacr.org/2004/331`.

[Bra93] Stefan A. BRANDS. An efficient off-line electronic cash system based on the representation problem. Technical report, Amsterdam, The Netherlands, The Netherlands, 1993.

[Bra94] Stefan BRANDS. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. STINSON, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, Heidelberg, August 1994.

[Bra97] Stefan BRANDS. Rapid demonstration of linear relations connected by Boolean operators. In Walter FUMY, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 318–333. Springer, Heidelberg, May 1997.

[BRT+10] Josep BALASCH, Alfredo RIAL, Carmela TRONCOSO, Bart PRENEEL, Ingrid VERBAUWHEDE, and Christophe GEUENS. Pretp : Privacy-preserving electronic toll pricing. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 5–5, Berkeley, CA, USA, 2010. USENIX Association.

[BS99] Mihir BELLARE and Amit SAHAI. Non-malleable encryption : Equivalence between two notions, and an indistinguishability-based characterization. In Michael J. WIENER, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 519–536. Springer, Heidelberg, August 1999.

[Cam98] Jan CAMENISCH. *Group signature schemes and payment systems based on the discrete logarithm problem*. PhD thesis, ETH Zurich, Zürich, Switzerland, 1998. URL : `http://d-nb.info/953066045`.

[Can01] Ran CANETTI. Universally composable security : A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CCJT13] Sébastien CANARD, Iwen COISEL, Amandine JAMBERT, and Jacques TRAORÉ. New results for the practical use of range proofs. In *Public Key Infrastructures, Services and Applications - 10th European Workshop, EuroPKI 2013, Egham, UK, September 12-13, 2013, Revised Selected Papers*, pages 47–64, 2013. URL : http://dx.doi.org/10.1007/978-3-642-53997-8_4, doi:10.1007/978-3-642-53997-8_4.

[CDHK15] Jan CAMENISCH, Maria DUBOVITSKAYA, Kristiyan HARALAMBIEV, and Markulf KOHLWEISS. Composable and modular anonymous credentials : Definitions and practical constructions. In Tetsu IWATA and Jung Hee CHEON, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48800-3_11.

[CGH98] Ran CANETTI, Oded GOLDREICH, and Shai HALEVI. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, 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*, volume 2951 of *LNCS*, pages 40–57. Springer, Heidelberg, February 2004.

[CH12] Jeremy CLARK and Urs HENGARTNER. Selections : Internet voting with over-the-shoulder coercion-resistance. In George DANEZIS, editor, *FC 2011*, volume 7035 of *LNCS*, pages 47–61. Springer, Heidelberg, February / March 2012.

[Cha81] David L. CHAUM. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2) :84–90, February 1981. URL : http://doi.acm.org/10.1145/358549.358563, doi:10.1145/358549.358563.

[Cha82] David CHAUM. Blind signatures for untraceable payments. In David CHAUM, Ronald L. RIVEST, and Alan T. SHERMAN, editors, *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, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Heidelberg, May 2005.

[CL01] Jan CAMENISCH and Anna LYSYANSKAYA. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit PFITZMANN, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

[CL03] Jan CAMENISCH and Anna LYSYANSKAYA. A signature scheme with efficient protocols. In Stelvio CIMATO, Clemente GALDI, and Giuseppe PERSIANO, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Heidelberg, September 2003.

[CL04] Jan CAMENISCH and Anna LYSYANSKAYA. Signature schemes and anonymous credentials from bilinear maps. In Matthew FRANKLIN, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

[CMZ14] Melissa CHASE, Sarah MEIKLEJOHN, and Greg ZAVERUCHA. Algebraic MACs and keyed-verification anonymous credentials. In Gail-Joon AHN, Moti YUNG, and Ninghui LI, editors, *ACM CCS 14*, pages 1205–1216. ACM Press, November 2014.

[Coh93] Henri COHEN. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, New-York, 1993.

[CS97] Jan CAMENISCH and Markus STADLER. Proof systems for general statements about discrete logarithms. Technical report, Citeseer, 1997.

[CS98] Ronald CRAMER and Victor SHOUP. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo KRAWCZYK, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, August 1998.

[CSS12] Kamalika CHAUDHURI, Anand SARWATE, and Kaushik SINHA. Near-optimal differentially private principal components. In F. PEREIRA, C. J. C. BURGES, L. BOTTOU, and K. Q. WEINBERGER, editors, *Advances in Neural Information Processing Systems 25*, pages 989–997. Curran Associates, Inc., 2012.

[Dam99] Ivan DAMGÅRD. *Commitment Schemes and Zero-Knowledge Protocols*, pages 63–86. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. URL : `https://doi.org/10.1007/3-540-48969-X_3`, `doi:10.1007/3-540-48969-X_3`.

[Des88] Yvo DESMEDT. Society and group oriented cryptography : A new concept. In Carl POMERANCE, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 120–127. Springer, Heidelberg, August 1988.

[DF90] Yvo DESMEDT and Yair FRANKEL. Threshold cryptosystems. In Gilles BRASSARD, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, August 1990.

[DH76] Whitfield DIFFIE and Martin E. HELLMAN. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976.

[DHKG11] Jeremy DAY, Yizhou HUANG, Edward KNAPP, and Ian GOLDBERG. SPEcTRe : spot-checked private ecash tolling at roadside. In *WPES*, pages 61–68. ACM, 2011.

[DKM+06] Cynthia DWORK, Krishnaram KENTHAPADI, Frank MCSHERRY, Ilya MIRONOV, and Moni NAOR. *Our Data, Ourselves : Privacy Via Distributed Noise Generation*, pages 486–503. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. URL : `https://doi.org/10.1007/11761679_29`, `doi:10.1007/11761679_29`.

[DKPW12] Yevgeniy DODIS, Eike KILTZ, Krzysztof PIETRZAK, and Daniel WICHS. Message authentication, revisited. In David POINTCHEVAL and Thomas JOHANSSON, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 355–374. Springer, Heidelberg, April 2012.

[DMNS06] Cynthia DWORK, Frank MCSHERRY, Kobbi NISSIM, and Adam SMITH. Calibrating noise to sensitivity in private data analysis. In Shai HALEVI and Tal RABIN, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 265–284. Springer, Heidelberg, March 2006.

[DR14] Cynthia DWORK and Aaron ROTH. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9 :211–407, August 2014.

[DTTZ14] Cynthia DWORK, Kunal TALWAR, Abhradeep THAKURTA, and Li ZHANG. Analyze gauss : Optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 11–20, New York, NY, USA, 2014. ACM. URL : `http://doi.acm.org/10.1145/2591796.2591883`, `doi:10.1145/2591796.2591883`.

[Dwo06] Cynthia DWORK. Differential privacy. In Michele BUGLIESI, Bart PRENEEL, Vladimiro SASSONE, and Ingo WEGENER, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006. `doi:10.1007/11787006_1`.

[DY05] Yevgeniy DODIS and Aleksandr YAMPOLSKIY. A verifiable random function with short proofs and keys. In Serge VAUDENAY, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, January 2005.

[ElG84] Taher ELGAMAL. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. BLAKLEY and David CHAUM, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[FFS87] Uriel FEIGE, Amos FIAT, and Adi SHAMIR. Zero knowledge proofs of identity. In Alfred AHO, editor, *19th ACM STOC*, pages 210–217. ACM Press, May 1987.

[FFS88] Uriel FEIGE, Amos FIAT, and Adi SHAMIR. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2) :77–94, 1988.

[FHS15] Georg FUCHSBAUER, Christian HANSER, and Daniel SLAMANIG. Practical round-optimal blind signatures in the standard model. In Rosario GENNARO and Matthew J. B. ROBSHAW, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Heidelberg, August 2015. `doi:10.1007/978-3-662-48000-7_12`.

[FP01] Pierre-Alain FOUQUE and David POINTCHEVAL. Threshold cryptosystems secure against chosen-ciphertext attacks. In Colin BOYD, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 351–368. Springer, Heidelberg, December 2001.

[FPV09] Georg FUCHSBAUER, David POINTCHEVAL, and Damien VERGNAUD. Transferable constant-size fair e-cash. In Juan A. GARAY, Atsuko MIYAJI, and Akira OTSUKA, editors, *CANS 09*, volume 5888 of *LNCS*, pages 226–247. Springer, Heidelberg, December 2009.

[FS87] Amos FIAT and Adi SHAMIR. How to prove yourself : Practical solutions to identification and signature problems. In Andrew M. ODLYZKO, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[FS01] Pierre-Alain FOUQUE and Jacques STERN. Fully distributed threshold RSA under standard assumptions. In Colin BOYD, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 310–330. Springer, Heidelberg, December 2001.

[GJKR03] Rosario GENNARO, Stanislaw JARECKI, Hugo KRAWCZYK, and Tal RABIN. Secure applications of Pedersen's distributed key generation protocol. In Marc JOYE, edi-

tor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 373–390. Springer, Heidelberg, April 2003.

[GK96] Oded GOLDREICH and Hugo KRAWCZYK. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1) :169–192, 1996.

[GM84] Shafi GOLDWASSER and Silvio MICALI. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2) :270–299, 1984.

[GMR85] Shafi GOLDWASSER, Silvio MICALI, and Charles RACKOFF. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[GMR89] Shafi GOLDWASSER, Silvio MICALI, and Charles RACKOFF. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1) :186–208, 1989.

[Gol04] Oded GOLDREICH. *Foundations of Cryptography : Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[GPS08] Steven D. GALBRAITH, Kenneth G. PATERSON, and Nigel P. SMART. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16) :3113–3121, 2008. URL : `http://dx.doi.org/10.1016/j.dam.2007.12.010`, `doi:10.1016/j.dam.2007.12.010`.

[GS08] Jens GROTH and Amit SAHAI. Efficient non-interactive proof systems for bilinear groups. In Nigel P. SMART, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[HR13] Moritz HARDT and Aaron ROTH. Beyond worst-case analysis in private singular vector computation. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 331–340, New York, NY, USA, 2013. ACM.

[HT07] Emeline HUFSCHMITT and Jacques TRAORÉ. Fair blind signatures revisited. In Tsuyoshi TAKAGI, Tatsuaki OKAMOTO, Eiji OKAMOTO, and Takeshi OKAMOTO, editors, *PAIRING 2007*, volume 4575 of *LNCS*, pages 268–292. Springer, Heidelberg, July 2007.

[IBM10] IBM. Specification of the identity mixer cryptographic library (revised version 2.3.0). IBM Research Report RZ 3730, 2010. `http://domino.research.ibm.com/library/cyberdig.nsf/1e4115aea78b6e7c85256b360066f0d4/eeb54ff3b91c1d648525759b004fbbb1?OpenDocument`.

[JCJ05] Ari JUELS, Dario CATALANO, and Markus JAKOBSSON. Coercion-resistant electronic elections. In *WPES*, pages 61–70, 2005.

[JCJ10] Ari JUELS, Dario CATALANO, and Markus JAKOBSSON. *Coercion-Resistant Electronic Elections*, pages 37–63. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. URL : `http://dx.doi.org/10.1007/978-3-642-12980-3_2`, `doi:10.1007/978-3-642-12980-3_2`.

[JJ00] Markus JAKOBSSON and Ari JUELS. Mix and match : Secure function evaluation via ciphertexts. In Tatsuaki OKAMOTO, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 162–177. Springer, Heidelberg, December 2000.

[JL00] Stanislaw JARECKI and Anna LYSYANSKAYA. Adaptively secure threshold cryptography : Introducing concurrency, removing erasures. In Bart PRENEEL, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242. Springer, Heidelberg, May 2000.

[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. URL : `http://dx.doi.org/10.1007/10722028_23`, `doi:10.1007/10722028_23`.

[JPBN09] L JEDRZEJCZYK, Blaine PRICE, Arosha BANDARA, and B NUSEIBEH. I know what you did last summer : risks of location data leakage in mobile and social computing. 01 2009.

[JS08] Tibor JAGER and Jörg SCHWENK. On the equivalence of generic group models. In Joonsang BAEK, Feng BAO, Kefei CHEN, and Xuejia LAI, editors, *ProvSec 2008*, volume 5324 of *LNCS*, pages 200–209. Springer, Heidelberg, October / November 2008.

[JSI96] Markus JAKOBSSON, Kazue SAKO, and Russell IMPAGLIAZZO. Designated verifier proofs and their applications. In Ueli M. MAURER, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 143–154. Springer, Heidelberg, May 1996.

[KL08] Jonathan KATZ and Andrew Y. LINDELL. Aggregate message authentication codes. In Tal MALKIN, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 155–169. Springer, Heidelberg, April 2008.

[KNRS13a] Shiva Prasad KASIVISWANATHAN, Kobbi NISSIM, Sofya RASKHODNIKOVA, and Adam SMITH. *Analyzing Graphs with Node Differential Privacy*, pages 457–476. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[KNRS13b] Shiva Prasad KASIVISWANATHAN, Kobbi NISSIM, Sofya RASKHODNIKOVA, and Adam SMITH. Analyzing graphs with node differential privacy. In Amit SAHAI, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 457–476. Springer, Heidelberg, March 2013. `doi:10.1007/978-3-642-36594-2_26`.

[Kob87] Neal KOBLITZ. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177) :203–209, 1987.

[KR96] Joe KILIAN and Phillip ROGAWAY. How to protect DES against exhaustive key search. In Neal KOBLITZ, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 252–267. Springer, Heidelberg, August 1996.

[KRSY11] Vishesh KARWA, Sofya RASKHODNIKOVA, Adam SMITH, and Grigory YAROSLAVTSEV. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11) :1146–1157, 2011.

[KSK14] Vishesh KARWA, Aleksandra B. SLAVKOVIĆ, and Pavel KRIVITSKY. *Differentially Private Exponential Random Graphs*, pages 143–155. Springer International Publishing, Cham, 2014. URL : `https://doi.org/10.1007/978-3-319-11257-2_12`, `doi:10.1007/978-3-319-11257-2_12`.

[KT13] Michael KAPRALOV and Kunal TALWAR. On differentially private low rank approximation. In *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 1395–1414, Philadelphia, PA, USA, 2013. Society for Industrial and Applied Mathematics. URL : `http://dl.acm.org/citation.cfm?id=2627817.2627918`.

[LHR+10] Chao LI, Michael HAY, Vibhor RASTOGI, Gerome MIKLAU, and Andrew MCGREGOR. Optimizing linear counting queries under differential privacy. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 123–134, New York, NY, USA, 2010. ACM. URL : `http://doi.acm.org/10.1145/1807085.1807104`, `doi:10.1145/1807085.1807104`.

[Lin16] Yehuda LINDELL. How to simulate it - A tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016. `http://eprint.iacr.org/2016/046`.

[LLV07] Ninghui LI, Tiancheng LI, and Suresh VENKATASUBRAMANIAN. t-closeness : Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 106–115. IEEE, 2007.

[LM14] Wentian LU and Gerome MIKLAU. Exponential random graph estimation under differential privacy. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 921–930, New York, NY, USA, 2014. ACM.

[LMRS04] Anna LYSYANSKAYA, Silvio MICALI, Leonid REYZIN, and Hovav SHACHAM. Sequential aggregate signatures from trapdoor permutations. In Christian CACHIN and Jan CAMENISCH, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 74–90. Springer, Heidelberg, May 2004.

[LMY15] Weiwei LIU, Yi MU, and Guomin YANG. An efficient privacy-preserving e-coupon system. In Dongdai LIN, Moti YUNG, and Jianying ZHOU, editors, *Information Security and Cryptology*, volume 8957 of *LCNS*, pages 3–15. Springer International Publishing, 2015. `doi:10.1007/978-3-319-16745-9_1`.

[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, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 465–485. Springer, Heidelberg, May / June 2006.

[LRSW99] Anna LYSYANSKAYA, Ronald L. RIVEST, Amit SAHAI, and Stefan WOLF. Pseudonym systems. In Howard M. HEYS and Carlisle M. ADAMS, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199. Springer, Heidelberg, August 1999.

[Mau05] Ueli M. MAURER. Abstract models of computation in cryptography (invited paper). In Nigel P. SMART, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

[McS09] Frank D. MCSHERRY. Privacy integrated queries : An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD In-*

ternational Conference on Management of Data, SIGMOD '09, pages 19–30, New York, NY, USA, 2009. ACM. doi:10.1145/1559845.1559850.

[MDNDD15] Milica MILUTINOVIC, Koen DECROIX, Vincent NAESSENS, and Bart DE DECKER. Privacy-preserving public transport ticketing system. In Pierangela SAMARATI, editor, *Data and Applications Security and Privacy XXIX*, volume 9149 of *LNCS*, pages 135–150. Springer International Publishing, 2015. doi:10.1007/978-3-319-20810-7_9.

[Mil86] Victor S. MILLER. Use of elliptic curves in cryptography. In Hugh C. WILLIAMS, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 417–426. Springer, Heidelberg, August 1986.

[MKGV07] Ashwin MACHANAVAJJHALA, Daniel KIFER, Johannes GEHRKE, and Muthuramakrishnan VENKITASUBRAMANIAM. l-diversity : Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1) :3, 2007.

[MM09] Frank MCSHERRY and Ilya MIRONOV. Differentially private recommender systems : Building privacy into the net. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 627–636, New York, NY, USA, 2009. ACM. URL : http://doi.acm.org/10.1145/1557019.1557090, doi:10.1145/1557019.1557090.

[MMCS11] Sarah MEIKLEJOHN, Keaton MOWERY, Stephen CHECKOWAY, and Hovav SHACHAM. The phantom tollbooth : Privacy-preserving electronic toll collection in the presence of driver collusion. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 32–32, Berkeley, CA, USA, 2011. USENIX Association. URL : http://dl.acm.org/citation.cfm?id=2028067.2028099.

[MVO91] Alfred MENEZES, Scott A. VANSTONE, and Tatsuaki OKAMOTO. Reducing elliptic curve logarithms to logarithms in a finite field. In *23rd ACM STOC*, pages 80–89. ACM Press, May 1991.

[MW09] D. J. MIR and R. N. WRIGHT. A differentially private graph estimator. In *2009 IEEE International Conference on Data Mining Workshops*, pages 122–129, Dec 2009. doi:10.1109/ICDMW.2009.96.

[MW12] Darakhshan MIR and Rebecca N. WRIGHT. A differentially private estimator for the stochastic kronecker graph model. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT '12, pages 167–176, New York, NY, USA, 2012. ACM.

[NAC07] Mehmet Ercan NERGIZ, Maurizio ATZORI, and Chris CLIFTON. Hiding the presence of individuals from shared databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 665–676. ACM, 2007.

[Nec94] V. I. NECHAEV. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2) :165–172, 1994.

[NRS07] Kobbi NISSIM, Sofya RASKHODNIKOVA, and Adam SMITH. Smooth sensitivity and sampling in private data analysis. In David S. JOHNSON and Uriel FEIGE, editors, *39th ACM STOC*, pages 75–84. ACM Press, June 2007.

[NS09] Arvind NARAYANAN and Vitaly SHMATIKOV. De-anonymizing social networks. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 173–187, Washington, DC, USA, 2009. IEEE Computer Society. URL : http://dx.doi.org/10.1109/SP.2009.22, doi:10.1109/SP.2009.22.

[NY90] Moni NAOR and Moti YUNG. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[OP01] Tatsuaki OKAMOTO and David POINTCHEVAL. The gap-problems : A new class of problems for the security of cryptographic schemes. In Kwangjo KIM, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer, Heidelberg, February 2001.

[Pai99] Pascal PAILLIER. Public-key cryptosystems based on composite degree residuosity classes. In Jacques STERN, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

[Paq13] Christian PAQUIN. U-Prove Technology Overview V1.1 (Revision 2). In Microsoft Technical Report, 2013. http://research.microsoft.com/pubs/166980/U-ProveTechnologyOverviewV1.1Revision2.pdf.

[PBB09] Raluca Ada POPA, Hari BALAKRISHNAN, and Andrew J. BLUMBERG. Vpriv : Protecting privacy in location-based vehicular services. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pages 335–350, Berkeley, CA, USA, 2009. USENIX Association. URL : http://dl.acm.org/citation.cfm?id=1855768.1855789.

[Ped92] Torben P. PEDERSEN. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan FEIGENBAUM, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[PH78] Stephen C. POHLIG and Martin E. HELLMAN. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Trans. Information Theory*, 24(1) :106–110, 1978. URL : http://dx.doi.org/10.1109/TIT.1978.1055817, doi:10.1109/TIT.1978.1055817.

[PHV14] Antonio DE LA PIEDRA, Jaap-Henk HOEPMAN, and Pim VULLERS. Towards a full-featured implementation of attribute based credentials on smart cards. In Dimitris GRITZALIS, Aggelos KIAYIAS, and Ioannis G. ASKOXYLAKIS, editors, *CANS 14*, volume 8813 of *LNCS*, pages 270–289. Springer, Heidelberg, October 2014. doi:10.1007/978-3-319-12280-9_18.

[Poi96] David POINTCHEVAL. *Les preuves de connaissance et leurs preuves de sécurité*. 1996.

[PS96] David POINTCHEVAL and Jacques STERN. Security proofs for signature schemes. In Ueli M. MAURER, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996.

[PS16] David POINTCHEVAL and Olivier SANDERS. Short randomizable signatures. In Kazue SAKO, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, February / March 2016. doi:10.1007/978-3-319-29485-8_7.

[PWWH16] John PAPARRIZOS, Ryen W WHITE, and Eric HORVITZ. Screening for pancreatic adenocarcinoma using signals from web search logs : Feasibility study and results. 12, 06 2016.

[PZ13] Christian PAQUIN and Greg ZAVERUCHA. U-Prove cryptographic specification V1.1 (Revision 3). In Microsoft Technical Report, 2013. http://research.microsoft.com/pubs/166969/U-ProveCryptographicSpecificationV1.1.pdf.

[RBHP15] Andy RUPP, Foteini BALDIMTSI, Gesine HINTERWÄLDER, and Christof PAAR. Cryptographic theory meets practice : Efficient and privacy-preserving payments for public transport. *ACM Trans. Inf. Syst. Secur.*, 17(3) :10 :1–10 :31, March 2015. URL : http://doi.acm.org/10.1145/2699904, doi:10.1145/2699904.

[RHBP13] Andy RUPP, Gesine HINTERWÄLDER, Foteini BALDIMTSI, and Christof PAAR. P4R : Privacy-preserving pre-payments with refunds for transportation systems. In Ahmad-Reza SADEGHI, editor, *FC 2013*, volume 7859 of *LNCS*, pages 205–212. Springer Heidelberg, 2013. doi:10.1007/978-3-642-39884-1_17.

[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, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[Sea16] Adam SEALFON. Shortest paths and distances with differential privacy. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '16, pages 29–41, New York, NY, USA, 2016. ACM. URL : http://doi.acm.org/10.1145/2902251.2902291, doi:10.1145/2902251.2902291.

[SGAW17] Ivana SEMANJSKI, Sidharta GAUTAMA, Rein AHAS, and Frank WITLOX. Spatial context mining approach for transport mode recognition from mobile sensed big data. *COMPUTERS ENVIRONMENT AND URBAN SYSTEMS*, 66 :38–52, 2017. URL : http://dx.doi.org/10.1016/j.compenvurbsys.2017.07.004.

[Sha69] Daniel SHANKS. Class number, a theory of factorization, and genera. In *Proceedings of Symposia in Pure Mathematics*, volume 20, pages 415–440, 1969.

[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, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[Sho04] Victor SHOUP. Sequences of games : a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/2004/332.

[SKHS12] Oliver SPYCHER, Reto E. KOENIG, Rolf HAENNI, and Michael SCHLÄPFER. A new approach towards coercion-resistant remote e-voting in linear time. In George DANEZIS, editor, *FC 2011*, volume 7035 of *LNCS*, pages 182–189. Springer, Heidelberg, February / March 2012.

[Swe98] Latanya SWEENEY. Protecting privacy when disclosing information : k-anonymity and its enforcement through generalization and suppression. Technical report, 1998.

[Swe02] Latanya SWEENEY. k-anonymity : A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5) :557–570, 2002. URL : http://dx.doi.org/10.1142/S0218488502001648, doi:10.1142/S0218488502001648.

[US 15] US Vote Foundation. End-to-end verifiable internet voting. In *The future of voting, Expert Statement*, 2015. URL : https://www.usvotefoundation.org/sites/default/files/E2EVIV_expert_statements.pdf.

[VA13] Pim VULLERS and Gergely ALPÁR. Efficient selective disclosure on smart cards using idemix. In Simone FISCHER-HÜBNER, Elisabeth DE LEEUW, and Chris MITCHELL, editors, *IDMAN 2013*, pages 53–67. Springer Berlin Heidelberg, 2013.

[WW13] Yue WANG and Xintao WU. Preserving differential privacy in degree-correlation based graph generation. *Trans. Data Privacy*, 6(2) :127–145, August 2013. URL : http://dl.acm.org/citation.cfm?id=2612167.2612168.

[WWW13] Yue WANG, Xintao WU, and Leting WU. *Differential Privacy Preserving Spectral Graph Analysis*, pages 329–340. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. URL : https://doi.org/10.1007/978-3-642-37456-2_28, doi:10.1007/978-3-642-37456-2_28.

[ZB11] Hui ZANG and Jean BOLOT. Anonymization of location data does not work : A large-scale measurement study. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 145–156, New York, NY, USA, 2011. ACM. URL : http://doi.acm.org/10.1145/2030613.2030630, doi:10.1145/2030613.2030630.

[ZXW+16] Matei ZAHARIA, Reynold S. XIN, Patrick WENDELL, Tathagata DAS, Michael ARMBRUST, Ankur DAVE, Xiangrui MENG, Josh ROSEN, Shivaram VENKATARAMAN, Michael J. FRANKLIN, Ali GHODSI, Joseph GONZALEZ, Scott SHENKER, and Ion STOICA. Apache spark : A unified engine for big data processing. *Commun. ACM*, 59(11) :56–65, October 2016. URL : http://doi.acm.org/10.1145/2934664, doi:10.1145/2934664.

Résumé

L'émergence de terminaux mobiles personnels, capables à la fois de communiquer et de se positionner, entraîne de nouveaux usages et services personnalisés. Néanmoins, ils impliquent une collecte importante de données à caractère personnel et nécessitent des solutions adaptées en termes de sécurité. Les utilisateurs n'ont pas toujours conscience des informations personnelles et sensibles qui peuvent être déduites de leurs utilisations. L'objectif principal de cette thèse est de montrer comment des mécanismes cryptographiques et des techniques d'anonymisation de données peuvent permettre de concilier à la fois le respect de la vie privée, les exigences de sécurité et l'utilité du service fourni.

Dans une première partie, nous étudions les accréditations anonymes avec vérification par clé. Elles permettent de garantir l'anonymat des utilisateurs vis-à-vis du fournisseur de service : un utilisateur prouve son droit d'accès, sans révéler d'information superflue. Nous introduisons des nouvelles primitives qui offrent des propriétés distinctes et ont un intérêt à elles-seules. Nous utilisons ces constructions pour concevoir trois systèmes respectueux de la vie privée : un premier système d'accréditations anonymes avec vérification par clé, un deuxième appliqué au vote électronique et un dernier pour le paiement électronique. Chaque solution est validée par des preuves de sécurité et offre une efficacité adaptée aux utilisations pratiques. En particulier, pour deux de ces contributions, des implémentations sur carte SIM ont été réalisées. Néanmoins, certains types de services nécessitent tout de même l'utilisation ou le stockage de données à caractère personnel, par nécessité de service ou encore par obligation légale.

Dans une seconde partie, nous étudions comment rendre respectueuses de la vie privée les données liées à l'usage de ces services. Nous proposons un procédé d'anonymisation pour des données de mobilité stockées, basé sur la confidentialité différentielle. Il permet de fournir des bases de données anonymes, en limitant le bruit ajouté. De telles bases de données peuvent alors être exploitées à des fins d'études scientifiques, économiques ou sociétales, par exemple.

Abstract

The emergence of personal mobile devices, with communication and positioning features, is leading to new use cases and personalized services. However, they imply a significant collection of personal data and therefore require appropriate security solutions. Indeed, users are not always aware of the personal and sensitive information that can be inferred from their use. The main objective of this thesis is to show how cryptographic mechanisms and data anonymization techniques can reconcile privacy, security requirements and utility of the service provided.

In the first part, we study keyed-verification anonymous credentials which guarantee the anonymity of users with respect to a given service provider : a user proves that she is granted access to its services without revealing any additional information. We introduce new such primitives that offer different properties and are of independent interest. We use these constructions to design three privacy-preserving systems : a keyed-verification anonymous credentials system, a coercion-resistant electronic voting scheme and an electronic payment system. Each of these solutions is practical and proven secure. Indeed, for two of these contributions, implementations on SIM cards have been carried out. Nevertheless, some kinds of services still require using or storing personal data for compliance with a legal obligation or for the provision of the service.

In the second part, we study how to preserve users' privacy in such services. To this end, we propose an anonymization process for mobility traces based on differential privacy. It allows us to provide anonymous databases by limiting the added noise. Such databases can then be exploited for scientific, economic or societal purposes, for instance.