Généralisation des protocoles en cas multi-utilisateurs

Laura Brouilhet

Généralisation des protocoles en cas multi-utilisateurs

Thèse

présentée et soutenue publiquement le 15 décembre 2020 par

Laura Brouilhet

pour l'obtention du

Doctorat de l'Université de Limoges
(spécialité mathématiques)

Devant le jury composé de :

Directeurs de thèse :	Olivier Blazy	Université de Limoges
	Duong Hieu Phan	Télécom Paris, Institut Polytechnique de Paris
Rapporteurs :	Pascal Lafourcade	Université Clermont Auvergne
	Ayoub Otmani	Université de Rouen
Examinateurs :	Céline Chevalier	Université Panthéon-Assas Paris 2
	Philippe Gaborit	Université de Limoges

Remerciements

Par avance, je tiens à m'excuser si j'oublie des personnes dans ces remerciements. Il s'agit d'un des exercices les plus difficiles mais également de la partie la plus lue dans une thèse.

Pour commencer de manière originale, je souhaiterai remercier mes directeurs Olivier Blazy et Duong-Hieu Phan. Je tiens à remercier particulièrement Olivier pour le temps qu'il a pris pour relire mes différents articles et les nombreuses versions de cette thèse. Merci également de m'avoir rassurer avant, pendant et après cette aventure de trois ans.

Je remercie également Pascal Lafourcade et Ayoub Otmani pour avoir accepté d'être les rapporteurs de cette thèse ainsi que Céline Chevalier et Philippe Gaborit pour participer à mon jury.

Je tiens à remercier mes co-auteurs Céline Chevalier, Emmanuel Conchon, Neals Fournaise, Mathieu Klingler, Patrick Towa, Ida Tucker et Damien Vergnaud, ce fut un plaisir de travailler à vos côtés.

Merci également à toutes les personnes avec lesquelles j'ai pu travailler au cours de ces trois dernières années au sein du laboratoire XLim : Benoît Crespin, Emmanuel Conchon, Damien Sauveron, Maxime Maria, Alain Salinier et Tristan Vaccon. Un grand merci à Débora Thomas et Sophie Queille pour faciliter notre travail et nos différentes démarches. Je remercie également Nancy Pieyre, Estelle Parrondo, Orélie Roget, Raymonde Besse, Christophe Durousseau et Anne Ménard pour leur accueil à l'école 3iL ainsi que leur confiance durant mes différents enseignements.

Je remercie les doctorants pour les pauses café mélangeant informatique, jeux vidéos et vidéos Youtube plus que douteuses : Nicolas, Maxime, Hamza, Neals et Mathieu. Je vous souhaite le meilleur pour la suite.

Merci à tout ceux avec qui j'ai eu la chance de faire mes études notamment Éric pour notre duo de blagues mais aussi pour toujours avoir un paquet de cartes au bon moment.

Je remercie Martine pour m'avoir accueillie à Limoges et pour toujours être à mes côtés quand j'en ai besoin.

Je remercie mes amis de toujours Kévin, Didier, Jérémie et Clara pour m'accueillir à chaque fois comme si je n'étais jamais partie.

Pour conclure ces remerciements, merci à mes parents d'y avoir toujours cru même quand ils étaient les seuls, de m'avoir appris à toujours se relever peu importe les épreuves. Enfin, merci à Maxime qui m'a soutenue, supportée, l'amour de ma vie.

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 Les débuts de la cryptographie

De tout temps les hommes ont souhaité cacher des secrets. Ces derniers peuvent être des secrets militaires (plan de bataille par exemple), des secrets économiques (des transactions bancaires), etc ... Cette volonté de dissimuler un message se retrouve dans l'étymologie de cryptographie : « $\kappa\rho\upsilon\pi\tau o\varsigma$ » qui signifie cacher et « $\gamma\rho\alpha\phi\epsilon\iota\nu$ » écrire. Une des plus vieilles traces retrouvées est l'inscription de ce qu'il semble être une recette sur une vieille poterie en Mésopotamie.

Au temps de l'Antiquité, on souhaitait seulement dissimuler le message alors que la cryptographie moderne a pour objectif d'assurer la *confidentialité*, l'*intégrité* et l'*authenticité* du message transmis.

Le principe de cacher le message dans un autre s'appelle la stéganographie. Il s'agissait de cacher le message en clair sur une personne ou un objet. La stéganographie était très utilisée au temps de l'antiquité comme le présente l'anecdote suivante.

Afin de prévenir Athènes d'une attaque imminente de la Perse, un exilé grec transmis une tablette de cire vierge à première vue. Au préalable, il avait gravé les informations puis les avait cachées en les recouvrant de cire. Ainsi, dès réception du message, il suffisait au destinataire de gratter la couche de cire pour retrouver le message. On peut également citer une autre technique de stéganographie : l'utilisation d'un être humain afin de dissimuler le message. Une personne grave le message sur le crâne d'un esclave et attend que ses cheveux repoussent. Ensuite, il lui suffit de l'envoyer au destinataire où ce dernier rase la tête du messager pour obtenir les informations. On peut constater que cette technique prend du temps pour être mise en œuvre.

Pour en revenir à l'évolution de la cryptographie, un des premiers chiffrements qui nous fut rapporté est le chiffrement de substitution dit de César. On lui attribue l'utilisation d'un protocole de chiffrement qui décale l'alphabet de trois lettres. Si l'on ne connaît pas ce décalage, le message semble incohérent. Ce chiffrement, attribué à Jules César, fut utilisé durant différentes guerres de l'empire romain afin de transmettre des messages. Même si le messager était intercepté par les ennemis, ces derniers ne pouvaient comprendre le message sans connaître le nombre de décalage nécessaire.

FIGURE 1.1 – Disque facilitant le chiffrement-déchiffrement de César.

Avec l'utilisation de chiffrement à des fins militaires, la cryptanalyse apparue afin d'essayer de casser ces chiffrements. Par casser un chiffrement, on entend le déchiffrer sans connaître la clé de chiffrement. Un des premiers traités exposant des techniques de cryptanalyse fut proposé par Al-Kindi au IX$^{\text{ième}}$ siècle dans *Manuscrit sur le déchiffrement des messages cryptographiques*. Al-Kindi explique comment, en utilisant le nombre d'apparition des lettres dans un texte clair, on peut déchiffrer un message chiffré. Cette technique s'appelle l'analyse de fréquences. Par exemple, en français, les lettres les plus utilisées sont le e, puis le a, ...

Durant le moyen-âge, la cryptographie continua de se développer dans les différentes cours d'Europe. Par exemple, la reine Marie d'Écosse fomenta l'assassinat de sa cousine la reine Élisabeth 1ère en utilisant des messages chiffrés. Cependant, les cryptanalystes de l'époque réussirent à le déchiffrer. Elle fut condamnée pour tentative de coup d'état et exécutée. Ainsi, la cryptographie n'était pas seulement utilisée en temps de guerre mais également lors de complot ou dès qu'un individu souhaitait transmettre des informations sensibles. Durant cette période se développa pour la première fois un chiffrement poly-alphabétique. Ces chiffrés furent présentés par Leon Battista Alberti en 1467. Un de ces chiffrements les plus connus est celui de Blaise de Vigenère qui résista aux attaques pendant plusieurs siècles. Un des avantages de ce chiffrement est sa résistance aux attaques par fréquence à la différence des chiffrements par substitution (celui de César par exemple).

À la fin du XIXième siècle, Auguste Kerckhoffs publia un essai militaire *La cryptographie militaire* dans lequel il présenta diverses propriétés que devaient vérifier un système cryptographique d'après lui. Pour la première fois, il est énoncé qu'un adversaire doit avoir accès aux systèmes, c'est-à-dire que la sécurité ne peut pas reposer sur la non-connaissance du système. On ne peut pas se contenter de dissimuler un secret pour que ce dernier soit protégé d'attaques. Le principe de Kerckhoffs que l'on peut déduire de ses écrits est : "la sécurité d'un cryptosystème ne doit reposer que sur le secret de la clé". De nos jours, ce principe est toujours suivi. Les différents principes énoncés par Kerckhoffs sont présentés dans la figure suivante.

1. Le système doit être matériellement, sinon mathématiquement indéchiffrable.
2. Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient, tomber dans les mains de l'ennemi.
3. La clé doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré de correspondants.
4. Il faut qu'il soit applicable à la correspondance télégraphique.
5. Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes.
6. Il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.

Les trois premières lois peuvent être considérées comme la base de la cryptographie moderne. Claude Shannon, à qui l'on doit la théorie de l'information, a reformulé la deuxième loi de la manière suivante : "l'adversaire connaît le système".

1.2 Le début de l'utilisation des machines

À la fin du XIXième siècle, les réseaux de télécommunications commencèrent à se développer. On peut notamment citer le télégraphe qui permet de transmettre un message à l'aide d'un câble ou encore la TSF (Télégraphie sans fil) qui permit le développement des transmissions radios. Les cryptographes de l'époque, depuis que Babbage et Kasiski avaient percé les secrets du chiffre de Vigenère, étaient à la recherche d'un nouveau chiffrement plus sûr. L'accélération du développement technologique ainsi que le début de la Grande Guerre, poussèrent les cryptographes à tenter de développer de nouveaux chiffrements. Après la première guerre mondiale et plusieurs échecs, les cryptographes se tournèrent vers des machines utilisant des disques à chiffrer.

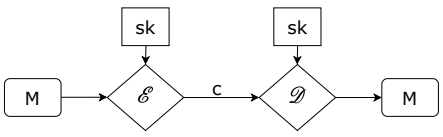
Un des exemples les plus connus de l'utilisation de machines en cryptographie se déroule durant la seconde guerre mondiale. L'armée allemande utilisait une machine cryptographique appelée machine Enigma. Les alliés avaient réussi à intercepter une machine utilisée par l'Allemagne nazie pour communiquer de manière chiffrée avec ses espions présents en Grande-Bretagne. L'enjeu fut d'essayer de comprendre comment fonctionner cette machine. En comprenant son fonctionnement, les alliés pourraient déchiffrer tous les messages qu'ils intercepteraient.

FIGURE 1.2 – Machine Enigma.

La machine Enigma fut inventée par Arthur Scherbius en 1918. Sa volonté était d'arrêter d'utiliser un crayon et du papier pour chiffrer un message, il souhaitait moderniser la cryptographie. Une première cryptanalyse fut menée contre la machine Enigma par une équipe de cryptologues polonais dont Marian Rejewski. Leurs travaux permirent de déchiffrer un grand nombre d'échanges entre les généraux de l'armée nazie. Une machine de cryptanalyse comprenant de nombreux retords fut installée à Bletchey Park, dans le centre de l'Angleterre. L'équipe chargée de "casser" Énigma était dirigé par Alan Turing. Son équipe put déchiffrer différents messages de l'armée allemande. La réussite de cette équipe permit de raccourcir la durée du conflit.

1.3 La cryptographie moderne

À la fin de la seconde guerre mondiale et avec la course à l'armement entre l'est et l'ouest, les nouvelles technologies ont connu un certain essor. C'est dans ce contexte mouvementé que le début des ordinateurs commença. Avec le commencement des recherches en informatique un nouveau problème apparu : comment sécuriser les informations que l'on veut transmettre en utilisant ces nouvelles machines ? Comme on a pu le voir précédemment, la cryptographie est une science millénaire qui a su s'adapter. Tous les principes cryptographiques précédents font partie de la cryptographie à clé privée ou cryptographie symétrique. Les utilisateurs doivent au préalable échanger une clé de chiffrement/déchiffrement pour pouvoir communiquer. Un des problèmes est la gestion du nombre de clés. On peut schématiser le chiffrement à clé privée sous la forme suivante :



À l'aide de la clé secrète sk, le message est chiffré par $\mathcal{E}$. Ainsi chiffré, il est déchiffré à l'aide de la même clé secrète.

1. Alice et Bob se mettent d'accord sur le chiffrement qu'ils utilisent.
2. Alice et Bob choisissent la clé en privée sk.
3. Alice chiffre le message m en utilisant sk.
4. Alice envoie le chiffré c à Bob.
5. Bob déchiffre c en utilisée la clé sk.

Avec l'évolution technologique, les chercheurs ont essayé de trouver une nouvelle méthode moins coûteuse en termes de nombre de clés mais ils ont également chercher à simplifier l'échange des clés. La question était de quelle manière est-il possible d'échanger une clé de manière sécurisée de l'autre côté du globe sans avoir à la rencontrer ?

En 1976, Whitfield Diffie et Martin Hellman proposèrent une solution : la cryptographie à clé publique ou cryptographie asymétrique. Dans leur article *New Directions in Cryptography* [DH76], ils jetèrent les bases de la cryptographie moderne qui ne cessa dès lors de s'améliorer et de se perfectionner. Dans la cryptographie à clé publique, à la différence de la cryptographie à clé privée, la clé de chiffrement et de déchiffrement sont différentes. La clé de chiffrement est publiquement connue. Le destinataire doit seulement posséder la clé secrète associée à cette clé publique pour être capable de déchiffrer le message. Ainsi, à la différence de la cryptographie à clé privée, il n'est plus difficile d'échanger la clé de déchiffrement.

1. Alice et Bob se mettent d'accord sur le chiffrement qu'ils utilisent.
2. Bob publie sa clé publique $\mathsf{pk}_{\mathcal{B}}$.
3. Alice chiffre le message m en utilisant $\mathsf{pk}_{\mathcal{B}}$.
4. Alice envoie le chiffré c à Bob.
5. Bob déchiffre c en utilisant sa clé privée usk_B.

Le concept important en cryptographie à clé publique est la considération d'une fonction $f : X \to Y$ que l'on peut facilement évaluer en tout point $x \in X$ mais dont l'inverse est difficile à calculer : pour un $y \in Y$ donné, il est difficile de trouver $x \in X$ tel que $y = f(x)$.

Le premier schéma de la cryptographie à clé publique est le protocole RSA. Il fut proposé par Ronald Rivest, Adi Shamir et Leonard Adleman en 1977 dans [RSA78]. Le protocole de chiffrement RSA repose sur la difficulté à pouvoir factoriser de grands nombres premiers. D'ailleurs, un concours mettait en compétition plusieurs nombres semi-premiers. Le but était de réussir à les factoriser. La génération de grands nombres premiers est un des enjeux majeurs de la cryptographie moderne. En théorie des nombres, la recherche de nombres premiers émergea dès l'antiquité. On peut notamment citer le crible d'Ératosthène qui date de la grèce antique. Depuis, plusieurs cribles furent proposés plus ou moins efficaces selon les versions. Par exemple, le crible d'Atkin proposé en 1999 par Atkin et Bernstein [AB04] est une version améliorée du crible d'Ératosthène. Comme la génération de grands nombres premiers est difficile, on peut utiliser en cryptographie des tests de primalité probabiliste. Le plus connu est le test de primalité de Fermat. Ce dernier est simple mais toutefois, il ne voit pas certains nombres composés. Comme test plus efficace on peut citer celui de Miller-Rabin ou celui de Solovay-Strassen. Ces tests probabilistes permettent de rendre la probabilité de primalité aussi proche de 1 que l'on souhaite.

En proposant de nouvelles primitives de chiffrement, les propriétés de sécurité ont également dû évoluer afin de s'assurer que les chiffrements utilisés étaient sûrs. On attend notamment d'un chiffrement à clé publique qu'il résiste aux attaques à texte clair choisis. Un adversaire en possession de plusieurs chiffrés associés à des messages ne peut pas obtenir d'informations sur la clé secrète.

Les protocoles de chiffrement à clé publique n'ont cessé de se développer depuis Diffie et Hellman. Des variantes de protocoles à clé publique existent notamment le chiffrement à base

d'identité. Le principe de ce chiffrement est simple. Plutôt que d'utiliser une suite de bit aléatoire comme clé de chiffrement, on utilise une clé en lien avec l'identité du destinataire. Par exemple, il peut s'agir de notre adresse mail, de notre numéro de sécurité sociale ou étudiant. Ce principe fut proposé par Shamir dans [Sha84]. Tout en étant pratique, ces protocoles sont difficiles à proposer. Les premiers protocoles de chiffrement reposants sur l'identité furent proposés en 2001.

Les protocoles de chiffrement ne sont pas les seules nouvelles primitives proposées par Diffie et Hellaman en 1976. Les signatures électroniques furent également introduites dans cet article. Ces signatures ont une fonction similaire aux signatures manuscrites : elles permettent d'authentifier le destinataire. Dans leur article présentant RSA, Rivest, Shamir et Adleman proposèrent également une signature. Après cet article plusieurs protocoles de signatures sont proposés, comme par exemple : celle de Lamport [Lam79] ou celle de Rabin [Rab79].

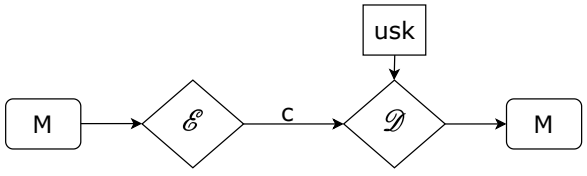
Un formalisme des propriétés de sécurité qu'une signature doit vérifier fut proposé par Goldwasser, Micali et Rivest dans [GMR88]. Ce formalisme permet d'assurer la sécurité d'un protocole de chiffrement au travers d'équations mathématiques.

Tout comme les protocoles de chiffrement, les protocoles de signatures n'ont cessé de se perfectionner pour répondre aux attentes du monde moderne. Par exemple, les signatures en blanc permettent d'effectuer une signature sur un message sans apprendre d'information sur ce dernier. Ces signatures peuvent par exemple être utilisées pour effectuer des transactions bancaires ou même pour le vote électronique.

De nos jours, la cryptographie est de plus en plus présente. Pendant plusieurs siècles, elle fut seulement utilisée à des fins militaires ou pour des intrigues de cours royales. La protection de la vie privée est un des enjeux majeurs de notre siècle. Avec l'avènements de différents réseaux sociaux ou des objets connectés (*Internet Of Things*) la sécurisation des données informatiques devient de plus en plus présente dans notre société. La cryptographie n'est plus seulement un art militaire, elle est devenue essentielle pour notre protection au quotidien que ce soit sur notre lieux de travail, dans nos loisirs ou même dans la sphère familiale. Dans cette thèse nous allons voir comment en utilisant des protocoles déjà existants on a pu en proposer de nouveaux qui permettent de s'adapter au nouveau challenge de la cryptographie : notre protection au quotidien.

1.4 Chiffrements et signatures de la cryptographie à clé publique

Dans cette thèse, nous allons nous focaliser sur des protocoles de chiffrement qui utilise l'identité du destinataire comme clé de chiffrement. Ces protocoles font partis de la cryptographie à clé publique. On peut par exemple imaginer une adresse mail ou le numéro de sécurité sociale pour tenir ce rôle car ils sont uniques. Le principe de ce chiffrement fut proposé par Adi Shamir [Sha84]. Ce protocole permet aux utilisateurs de transmettre des messages sans communiquer au préalable de clé de chiffrement comme on peut le voir dans le schéma suivant. L'utilisateur peut déchiffrer le chiffré en utilisant sa clé associée à son identité.



Un des problèmes de ces protocoles est la difficulté à en proposer. Il a fallu attendre 2001 pour voir apparaître de premières instanciations notamment celles de Boneh-Franklin [BF03] et de Cocks [Coc01a].

Les IBE font partie d'une plus grande famille de protocoles : les chiffrements par attributs (ABE). Paradoxalement, les ABE sont apparus après les IBE. Ils furent présenter par Sahai et Waters [SW05] comme une généralisation des IBE. Dans ces chiffrements, le destinataire doit

vérifier certains critères pour pouvoir déchiffrer. Ces critères sont appelés attributs. Par exemple, si un message envoyé ne peut être lu que par des doctorants de l'Université de Limoges, un doctorant de l'Université d'Aix-Marseille ne pourra pas le lire. Il est bien un doctorant mais pas dans la bonne université. D'un point de vue plus technique, on traduit ces propriétés par des formules binaires utilisant des XOR et/ou des OR.

On attend des IBE qu'ils respectent différentes propriétés de sécurité (développées dans le chapitre 2, section 2.2.1). L'identité du destinataire du message doit être protégée contre un adversaire malveillant. Un adversaire ne doit pas savoir pour quelle identité un message est chiffré. Si c'est le cas, on dit que notre IBE est anonyme (ANON-ID-CPA). De plus, un adversaire ne doit pas pouvoir savoir à quel message est associé un chiffré. Dans ce cas, un IBE est sémantiquement sûr (IND-ID-CPA).

En cryptographie, les protocoles de chiffrement ne sont pas les seules primitives même s'il s'agit de celles qui sont le plus connues du grand public. Il existe également les signatures électroniques qui permettent d'assurer l'authenticité d'un message. Les signatures électroniques sont analogues aux signatures manuscrites : elles permettent de certifier l'authenticité d'un message.

Tout comme pour les protocoles de chiffrement, on attend qu'un protocole de signature soit sûr *i.e.* qu'il assure l'intégrité du message même si un adversaire l'intercepte. Pour les signatures, la propriété analogue à l'IND-ID-CPA des IBE est la résistance aux contrefaçons. Un adversaire, tout en ayant n messages signés en sa possession ne peut pas générer une $n + 1^{\text{ième}}$ signature valide. Cette propriété permet d'assurer l'authenticité du message.

De la même manière que pour les protocoles il existe plusieurs types de signatures électroniques. Dans ce manuscrit, nous allons notamment étudier les signatures en blanc (*blind signature*) ainsi que les signatures par attributs (ABS).

Les signatures en blanc furent proposées par Chaum [Cha82] en 1982. L'intérêt de ces signatures est d'éviter le traçage des échanges bancaires. Dans cette signature, le signataire (souvent le signataire est un serveur) signe le message mais ne peut pas lire ce dernier. Avant de transmettre le message au signataire, on peut imaginer que l'utilisateur le masque en utilisant un protocole de chiffrement tel que celui de ElGamal. Concernant la sécurité, une signature en blanc doit résister aux contrefaçons mais également être *blind*. Il faut bien vérifier que l'adversaire possédant deux signatures, ne peut pas savoir pour quel message la signature a été générée.

Une autre variante de signature est les signatures par attributs (ABS). Un des intérêts de cette signature est de pouvoir permettre à certaines personnes de vérifier la signature. Ainsi, on peut désigner des personnes pour la vérifier sans connaître leur identité. On peut par exemple, permettre à tous les médecins de Limoges de vérifier la signature sans pour autant connaître leur noms. Toute comme un protocole classique, une ABS doit résister aux contrefaçons mais également protéger la vie privée du signataire.

Afin de construire ces différents protocoles plusieurs outils peuvent être utilisés. Dans ce manuscrit, nous allons notamment utiliser des *smooth projective hash function* (SPHF) et des preuves non interactives à divulgation nulle de connaissance (NIZK). Les SPHF ont notamment permis de construire des protocoles avec une sécurité plus forte. Les NIZK sont utilisés dans ce manuscrit afin de proposer une signature en blanc en tour optimal et de taille constante.

1.5 Contributions

Les travaux effectués dans le cadre de cette thèse ont permis la rédaction de plusieurs articles dont les résumés sont présentés ci-après.

— *Anonymous Identity-based Encryption with Traceable identity* (AIBET)

Olivier Blazy et Duong Hieu Phan

Kiayias *et al.* [KTY04] ont proposé des signatures de groupes traçables. De manière analogue, nous avons proposé une modification d'IBE afin de pouvoir tracer l'identité du

destinataire. Dans notre construction, en plus des algorithmes classiques d'un IBE (Setup, KeyGen, Encrypt et Decrypt), nous avons ajouté un algorithme TSKGen qui génère une clé de traçage tsk pour l'identité recherchée. Pour savoir si l'identité recherchée est celle utilisée du destinataire du message, nous utilisons une équation de couplages. Si les deux identités sont identiques, l'algorithme de vérification renvoie 1. De plus, comme il est possible de le voir sur les deux transformations que nous avons effectuées sur des IBE existants, l'ajout de deux algorithmes TSKGen et TVerify ne modifie pas les propriétés de sécurité des protocoles.

Nous avons proposé deux applications de cette modification à des IBE pré-existants ([BW06, BKP14]) tout en conservant des hypothèses de sécurité standards. Nos transformations conservent l'anonymat et la sécurité sémantique.
Travail publié et présenté à la conférence ARES 2019 [BBP19].

— *Round Optimal Blind Signature*

Olivier Blazy, Céline Chevalier et Neals Fournaise

Les signatures en blanc ou *blind signatures* sont connues depuis plusieurs décennies en cryptographie. Toutefois, depuis leur présentation par Chaum en 2001, plusieurs auteurs ont proposé des signatures en blanc sans être en tour optimal sous des hypothèses de sécurité classiques. C'est en souhaitant trouver une solution à ce problème ouvert que nous avons proposé une signature sur un chiffré randomisable que nous utilisons pour construire notre *Round Optimal Blind Signature* en utilisant la méthode de Fischlin [Fis06]. Leur sécurité repose sur une hypothèse classique : SXDH.
Dans cet article, nous présentons d'abord une signature SRC qui peut notamment être appliquée au vote électronique. À partir de cette signature, nous avons pu construire notre signature en blanc.
Travail publié à la conférence SECRYPT 2020 [BBCF20].

— *Attribute-based Designated Verifier Signature* (ABDVS)

Olivier Blazy, Emmanuel Conchon et Mathieu Klingler

En cryptographie, ils existent différentes sortes de signatures électroniques qui permettent de résoudre des problèmes au sein d'entreprises par exemple. Ainsi, la signature présentée dans cet article combine les propriétés de plusieurs signatures : les attributs pour désigner un groupe de vérifieurs et des vérifieurs désignés ce qui permet d'éviter les fuites d'informations. On peut imaginer des cas de figures où un groupe de personnes ayant les mêmes caractéristiques peut être autorisé à vérifier une signature *e.g.* les médecins d'une ville ou des enseignants. Notre signature, notée ABDVS est construite sur le modèle de la signature de Waters. On utilise un schéma de chiffrement basé sur l'identité afin de masquer la signature. La vérification de la signature est effectuée à l'aide d'une équation de couplage. Une signature qui désigne un vérifieur (DVS) permet seulement à la personne qui est désignée de vérifier la signature sans pouvoir fournir d'informations supplémentaires à une tierce personne. Notre construction utilise plusieurs briques de cryptographie et vérifie les différentes hypothèses de sécurité des signatures à base d'attributs et des DVS sous des hypothèses classiques.
En cours de soumission.

— *Hardware security without secure hardware : How to sign with a password and a server ?*

Olivier Blazy, Céline Chevalier, Patrick Towa, Ida Tucker et Damien Vergnaud

Actuellement, l'utilisation d'appareils connectés, d'ordinateurs est devenue une habitude quotidienne. La signature proposée dans cet article s'inscrit dans une volonté de protéger l'utilisateur de serveur et d'appareils connectés qui pourraient être malicieux. Nous souhaitions permettre à un utilisateur d'obtenir une signature en se protégeant d'un serveur malveillant. Dans notre modèle, l'appareil connecté, appelé token, et le serveur ne

peuvent pas signer l'un sans l'autre. De plus, la signature est une signature en blanc : ils ne peuvent pas obtenir d'information sur le message. Cette propriété permet de protéger l'utilisateur. Étant également résistante aux attaques, notre signature protège les signataires d'un individu malveillant. Dans l'instanciation associée à cette nouvelle primitive appelée TASS, nous utilisons notamment une signature de Waters asymétrique ainsi que des *smooth projective hash functions*. La sécurité de cette construction repose sur une hypothèse classique : CDH+.

Un protocole de chiffrement associé est en cours de soumission à la conférence PKC 2021.

Les différentes notions présentées dans les premiers chapitres de cette thèse sont utilisées afin de proposer soit de nouvelles primitives soit des améliorations de primitives préexistantes.

Le schéma ci-dessous permet de mettre en lumière ces relations. On peut par exemple voir qu'à partir d'un protocole de signatures et de preuves NIZK, on a pu proposer une signature en blanc.

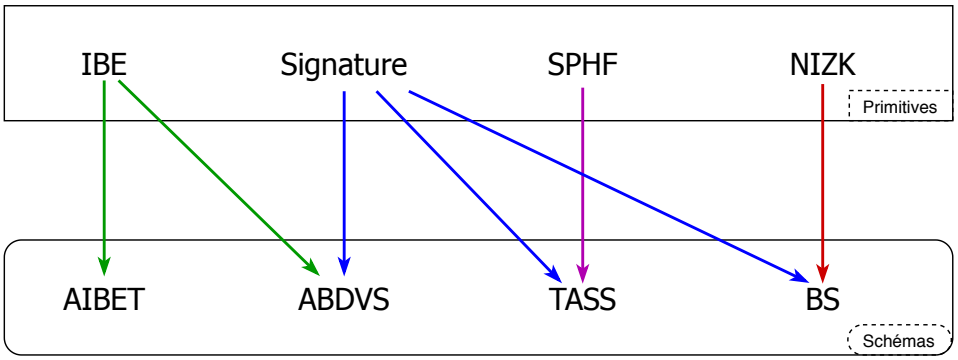


FIGURE 1.3 – Relations entre différentes primitives cryptographiques et les schémas présentés dans ce manuscrit.

Afin de présenter ces différentes constructions, nous allons dans un premier temps énonces des définitions classiques en cryptographie à clé publique tels que le chiffrement ou les signatures électroniques ainsi que des protocoles utilisés dans la suite du manuscrit.

Ensuite, sont présentées des signatures dont les applications s'inscrivent dans l'actualité. La première, une signature qui désigne des vérifieurs à partir de leurs attributs peut s'appliquer au milieu médicale. Les secondes, une signature sur message chiffré *randomizable* et une signature en blanc, peuvent s'appliquer au vote électronique ou aux monnaies électroniques. Nous allons voir que ces différentes signatures, par leur propriété permettent de protéger la vie privée de l'utilisateur mais également le signataire d'un faussaire.

Après avoir étudié ces signatures, nous allons nous concentrer sur un protocole de chiffrement. Ce dernier permet de savoir si l'identité du destinataire est une identité que l'on recherche. Faire ce test ne permet pas d'obtenir des informations sur le message *i.e.* la personne qui effectue le test ne peut pas déchiffrer en utilisant la clé de traçage.

Pour finir cette thèse, une signature entre trois parties est présentée. L'avantage de cette signature est que nous n'avons pas d'hypothèse sur une des parties. La clé de signature est partagée entre le serveur et le token. Par conséquent, on attend qu'aucune des parties ne puissent signer sans l'autre.

Table de notation

Afin de faciliter la lecture de ce manuscrit, voici une liste regroupant les différentes notations qui sont utilisées.

- $\mathcal{A}$: l'adversaire dans nos jeux de sécurité.
- Adv : l'avantage de l'adversaire $\mathcal{A}$.
- $\mathcal{B}$: le simulateur dans nos jeux de sécurité.
- $\mathcal{C}$: l'espace des chiffrés.
- CT : liste des chiffrés utilisés dans un jeu de sécurité.
- c : un chiffré appartenant à $\mathcal{C}$.
- dk : la clé (privée) de déchiffrement.
- $\mathcal{G}$: groupe bilinéaire de la forme : $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$
- $\mathbb{G}$: un groupe d'ordre p.
- e : un couplage (une forme bilinéaire non-dégénérée).
- ek : la clé (publique) de chiffrement
- g : un générateur du groupe $\mathbb{G}$.
- $\mathfrak{K}$: le paramètre de sécurité.
- $\mathcal{M}$: l'espace des messages.
- m : un message de $\mathcal{M}$.
- mpk : clé maîtresse publique.
- msk : clé maîtresse privée.
- $\mathbb{N}$: l'ensemble des entiers naturels.
- p : un grand nombre premier.
- param : les paramètres globaux du système.
- *ppt* : *probabilistic polynomial time*.
- $\mathcal{Q}_{\mathsf{ID}}$: ensemble des identités ayant fait l'objet d'une requête par $\mathcal{A}$ dans un jeu de sécurité.
- $\mathcal{S}$: l'espace des signatures.
- $\mathcal{SM}$: l'espace des messages signés.
- sk : la clé de signature.
- $\mathcal{U}$: l'utilisateur du protocole.
- $\mathbb{U}$: l'univers des attributs.
- vk : la clé de vérification d'une signature.
- $\mathbb{Z}$: le groupe des entiers relatifs.
- $\mathbb{Z}_p$: le groupe cyclique des entiers relatifs $\mathbb{Z}$ modulo p.
- σ : une signature appartenant à $\mathcal{S}$.
- * : est associé au message ou à l'identité challenge (m^*, id^*).

- $\xleftarrow{\$}$: signifie que l'on choisit un élément de manière aléatoire.

Nous utilisons la notation de [GS08] suivante pour les opérations sur les vecteurs :

— $\cdot$ pour le produit matriciel classique.

— $\odot$ pour le produit coordonnées à coordonnées.

— $\mathbf{A}||\mathbf{B}$ la concaténation de deux matrices ayant le même nombre de lignes (où $a||b+c$ est l'écriture simplifiée de $a||(b+c)$).

Pour une matrice $\mathbf{A} \in \mathbb{Z}_p^{(k+1)\times n}$, $\overline{\mathbf{A}} \in \mathbb{Z}_p^{k\times n}$ correspond à la partie supérieure de la matrice $\mathbf{A}$ et $\underline{\mathbf{A}} \in \mathbb{Z}_p^{1\times n}$ la dernière colonne de la matrice $\mathbf{A}$.

Glossaire

Cette thèse étant rédigée en français, plusieurs termes cryptographiques ont été traduits. Afin de familiariser le lecteur avec ces nouveaux termes, voici un glossaire qui les recense.

— *Attribute-based Signature* : Signature par attributs.

— *Blind Signature* : Signature en blanc.

— *Commit* : Mise en gage.

— *Identity-based encryption* (IBE) : Chiffrement basé sur l'identité.

— *Message Authentification Code* (MAC) : Code d'authentification du message.

— *Non-interactive zero-knowledge proofs* (NIZK) : preuves non-interactives à divulgation nulle de connaissance.

— *Threshold Signing* : Signature à seuil.

— *Unforgeability* : Résistance aux contrefaçons.

— *Universal one-way function* : Fonction universelle à sens unique.

Chapitre 2

Définitions

Dans ce chapitre nous allons présenter les différentes notions essentielles utilisées dans la suite de ce manuscrit. Ce chapitre est articulé en trois parties.

À la différence de celle utilisée durant l'Antiquité, la cryptographie moderne utilise les mathématiques afin de vérifier certaines conditions comme l'authenticité, la confidentialité ou l'intégrité d'un message échangé. Dans ce sens, la première partie des définitions présente différents problèmes mathématiques utilisés dans nos protocoles de chiffrement et de signature. Nous présentons également des notions importantes en cryptographie telles que les fonctions de hachage ou la mise en gage (section 2.1.3).

Dans la seconde partie, nous présenterons un des points, si ce n'est le point le plus important en cryptographie : le chiffrement. Comme nous avons pu le voir dans l'introduction, la volonté de cacher des secrets fut toujours présente chez l'Homme. Toutefois, ici, nous allons nous attacher à présenter la notion de chiffrement cryptographique sous forme de définition ainsi que les propriétés de sécurité que nous souhaitons qu'il vérifie. Plusieurs variantes des protocoles de chiffrement existent comme par exemple le chiffrement basé sur l'identité. Cette notion est développée dans la suite de ce chapitre car elle est utilisée afin de construire des protocoles de signature et de chiffrement.

La troisième et dernière partie de ce chapitre est axée sur les signatures électroniques. De manière analogue au chiffrement, nous énonçons en premier la définition d'une signature ainsi que les propriétés de sécurité qu'elle doit vérifier. Tout comme pour le chiffrement, des variantes de la signature existent afin de s'appliquer à différentes problématiques.

Sommaire

2.1 Définitions Générales

Cette section comprend différentes définitions mathématiques que nous utilisons essentiellement afin de prouver la sécurité de protocoles cryptographiques. Dans la suite, nous appelons ces définitions des problèmes difficiles. On entend par là, qu'il n'existe pas d'algorithme efficace capable de les résoudre.

2.1.1 Les problèmes difficiles

Comme nous l'avons vu en introduction, afin de garantir la sécurité des données, la cryptographie cherche à assurer leur confidentialité, leur authenticité et leur intégrité. Afin de garantir si un protocole de chiffrement ou de signature vérifient ces propriétés, les chercheurs se sont appuyés sur des problèmes reconnus comme difficiles d'un point de vue mathématiques. Ainsi si un attaquant arrivait à casser un tel protocole, cela reviendrait à dire qu'il a réussi à résoudre un de ces problèmes difficiles. À l'heure actuelle, nous n'avons pas les capacités pour résoudre ces problèmes dont certains sont connus depuis plusieurs années. C'est donc en nous appuyant sur de tels problèmes que nous proposons nos constructions cryptographiques.

Afin de formaliser cette idée, nous allons présenter différentes définitions d'hypothèses de sécurité que nous utilisons dans la suite de ce manuscrit.

Définition 1 (Algorithme en temps polynomial)**.** *Un algorithme est en temps polynomial si sa complexité est un $\mathcal{O}(n^k)$ avec n la taille des données qu'il prend en entrée et $k \in \mathbb{N}$.*

Définition 2 (Problème Difficile)**.** *Une fonction $f : \mathbb{N} \to \mathcal{R}$ est dite négligeable si $\forall c \in \mathbb{N}, \exists k_0 \in \mathbb{N}$ tel que $\forall k \geq k_0 : |f(k)| < k^{-c}$. Un problème est dit difficile s'il n'existe pas d'algorithme en temps polynomial qui le résout avec une probabilité non négligeable.*

Définition 3 (Problème du logarithme discret)**.** *Soit $\mathbb{G}$ un groupe d'ordre p. Connaissant $(p, \mathbb{G}, g)$ et un élément de groupe $h \in \mathbb{G}$ il est difficile de trouver μ sachant $h = g^\mu$.*

Définition 4 (Groupe Bilinéaire)**.** *Un groupe bilinéaire est de la forme $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ avec $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ des groupes cycliques d'ordre un grand nombre premier p dont les générateurs sont g_1, g_2 et $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ est une forme bilinéaire non-dégénérée,* i.e. *:*

$$\forall x \in \mathbb{G}_1, \forall y \in \mathbb{G}_2, \forall \lambda, \mu \in \mathbb{Z}_p : e(x^\lambda, y^\mu) = e(x, y)^{\lambda\mu}$$

Remarque 1. *$\mathbb{G}_T$ est appelé le* target group i.e. *le groupe cible.*

Définition 5 (Decisional Diffie-Hellman (DDH)[Bon98])**.** *Soit $\mathbb{G}$ un groupe d'ordre p premier. L'hypothèse* DDH *explique que pour un tuple $(g, g^a, g^b, g^c) \in \mathbb{G}$ donné, il est difficile de savoir quand $c = ab$.*

Les deux définitions suivantes sont des variantes de l'hypothèse DDH qui sont utilisées dans des groupes bilinéaires.

Définition 6 (External Diffie-Hellman (XDH [BBS04]))**.** *Il existe deux groupes $\mathbb{G}_1, \mathbb{G}_2$ tels que même si* DDH *est facile dans le groupe $\mathbb{G}_2$ alors* DDH *est difficile dans le groupe $\mathbb{G}_1$.*

Définition 7 (Symmetric External Diffie-Hellman (SXDH [ACHdM05]))**.** *Cette variante de* DDH *précise que s'il n'existe pas de morphisme de groupe efficace entre deux groupes bilinéaires $\mathbb{G}_1$ et $\mathbb{G}_2$, alors* DDH *est difficile à résoudre dans les deux groupes.*

Définition 8 (Decisional Linear Assumption (DLin [BBS04]))**.** *Dans un groupe multiplicatif $(p, \mathbb{G}, g)$ avec $(g^\lambda, g^\mu, g^{\alpha\lambda}, g^{\beta\mu}, g^\psi)$ pour $\alpha, \beta, \lambda, \mu \xleftarrow{\$} \mathbb{Z}_p$ inconnues aléatoires, il est difficile de savoir quand $\psi = \alpha + \beta$.*

Définition 9 (Computational Diffie-Hellman(CDH) [DH76]). *Dans un groupe multiplicatif $(p, \mathbb{G}, g)$, pour $\lambda, \mu \leftarrow \mathbb{Z}_p$ inconnus mais connaissant g^λ, g^μ, il est difficile de calculer $g^{\lambda\mu}$.*

Définition 10 (Advanced Computational Diffie-Hellman problem (CDH+)[BFPV11]). *Soit $\mathbb{G}_1, \mathbb{G}_2$ deux groupes multiplicatifs d'un ordre un grand nombre premier p dont g_1 et g_2 sont leurs générateurs respectifs. Soit $\lambda, \mu \leftarrow \mathbb{Z}_p$, connaissant $(g_1, g_2, g_1^\lambda, g_1^\mu, g_2^\lambda)$, il est difficile de calculer $g_1^{\lambda\mu}$*

À l'aide de ces problèmes difficiles nous pourrons définir des algorithmes cryptographiques. Afin de vérifier si ces derniers sont toujours sûrs nous pourrons alors nous appuyer sur des jeux de sécurité au cours desquels on utilise la notion d'avantage d'un adversaire.

Définition 11. *L'avantage d'un adversaire $\mathcal{A}$ de gagner un jeu G est la probabilité que $\mathcal{A}$ gagne ce jeu moins 1/2. On note cette notion* Adv.

Notre objectif est de montrer que l'on peut majorer cet avantage par une quantité très petite. Toutefois, si ce n'est pas possible, cela revient à dire que l'adversaire peut résoudre facilement un problème difficile.

Les définitions précédentes vont nous servir de base afin de pouvoir introduire mais également construire différents protocoles cryptographiques.

Diffie-Hellman sous forme matricielle

Dans la suite de cette thèse, nous allons utiliser la représentation implicite des éléments de groupes afin de simplifier la lecture lors, notamment, de l'utilisation simultanée de plusieurs groupes. Cette notation fut introduite par Escala *et al.* dans [EHK+13].

Considérons un algorithme probabiliste en temps polynomial GGen qui, prenant en entrée $1^\mathfrak{K}$ renvoie un groupe bilinéaire $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$.

Définition 12 (Représentation Implicite). *Pour $s \in \{1, 2, T\}$ et $a \in \mathbb{Z}_p$ on définit $[a]_s = g_s^a \in \mathbb{G}_s$ comme la représentation implicite de a dans le groupe $\mathbb{G}_s$. Si on travaille avec un unique groupe $\mathbb{G}$, on pose $[a] = g^a \in \mathbb{G}$.*

En s'appuyant sur la définition précédente, on peut étendre la notation aux matrices $\mathbf{A}$ de la forme : $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{m\times n}$. On définit $[\mathbf{A}]_s$ comme la représentation implicite de la matrice $\mathbf{A}$ dans le groupe $\mathbb{G}_s$:

$$[\mathbf{A}]_s := \begin{pmatrix} g_s^{a_{11}} & \dots & g_s^{a_{1m}} \\ & & \\ g_s^{a_{n1}} & \dots & g_s^{a_{nm}} \end{pmatrix} \in \mathbb{G}_s^{n\times m}$$

Ainsi, avec ces notations, on retrouve :

— le problème du logarithme discret : connaissant $[a]_s \in \mathbb{G}_s$ il est difficile de retrouver la valeur de a.

— le problème d'inversion de couplage : connaissant $[b]_T \in \mathbb{G}_T$ est difficile de retrouver $[b]_1 \in \mathbb{G}_1$ et $[b]_2 \in \mathbb{G}_2$.

Définition 13 (Notation du couplage). *Pour $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$, on définit $e([\mathbf{a}]_1, [\mathbf{b}]_2) := [\mathbf{a}^\top \mathbf{b}]_T \in \mathbb{G}_T$.*

Pour un $[a]_s \in \mathbb{G}_s$ donné et un scalaire $x \in \mathbb{Z}_p$, il est facile de calculer $[ax]_s \in \mathbb{G}_s$. De même, avec $[a]_1, [b]_2$ donnés, il est facile de calculer $[ab]_T$ en utilisant le couplage e.

Plus haut, nous avons présenté différentes définitions de problèmes utilisés dans la suite. En connaissant la notation matricielle, nous pouvons présenter différents problèmes utilisant ces notations.

Définition 14 (Distribution Matricielle [EHK+13]). *Soit $k \in \mathbb{N}$. On dit que $\mathcal{D}_k$ est une distribution matricielle s'il renvoie une matrice de rang maximal dans $\mathbb{Z}_p^{(k+1)\times k}$ en temps polynomial.*

Définition 15 ($\mathcal{D}_k$-Matrix Diffie-Hellman Assumption ($\mathcal{D}_k - \mathsf{MDDH}$) [EHK+13]). *Soit $\mathcal{D}_k$ une distribution matricielle et $s \in \{1, 2, T\}$. On dit que l'hypothèse $\mathcal{D}_k$-MDDH tient dans le groupe $\mathbb{G}_s$ si pour tout adversaire $\mathcal{A}$ en temps polynomial :*

$$\mathsf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_k,\mathsf{Setup}}(\mathcal{A}) := |\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{Aw}]_s) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{u}]_s)] = 1| = \mathsf{negl}(\mathfrak{K}),$$

où $\mathcal{G} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

Définition 16 ($\mathcal{D}_k$-Kernell Diffie-Hellman Assumption ($\mathcal{D}_k$-KerMDDH) [MRV15]). *Soit $\mathcal{D}_k$ une distribution matricielle et $s \in \{1, 2\}$.*

On dit que l'hypothèse $\mathcal{D}_k$-KerMDDH tient dans le groupe $\mathbb{G}_s$ si pour tout adversaire $\mathcal{A}$ en temps polynomial :

$$\mathsf{Adv}^{\mathsf{KerMDDH}}_{\mathcal{D}_k,\mathsf{Setup}}(\mathcal{A}) := |\Pr[\boldsymbol{c}\mathbf{A} = 0 \wedge \boldsymbol{c} \neq 0 | [\boldsymbol{c}]_{3-s} \xleftarrow{\$} \mathcal{A}(\mathcal{G}, [\mathbf{A}]_s)] = \mathsf{negl}(\mathfrak{K}),$$

où $\mathcal{G} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$ et $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$.

2.1.2 Modèles et Preuves de sécurité

En ce qui concerne la sécurité d'un protocole, il y a deux notions à ne pas confondre. Un protocole de chiffrement peut être reconnu sûr au sens de la théorie de l'information si l'adversaire ne peut pas obtenir, tirer d'informations des messages échangés. On peut également dire qu'un protocole de chiffrement est sûr d'un point de vue calculatoire, si l'adversaire, ayant en sa possession tous les messages échangés, ne peut pas obtenir des informations de ces derniers sans résoudre un problème difficile.

Dans ce manuscrit, nous nous concentrons sur la sécurité calculatoire des protocoles en utilisant les problèmes difficiles de la partie précédente.

Afin de démontrer la sécurité de nos algorithmes, nous appliquons nos hypothèses de sécurité dans différents modèles. Nous allons en présenter les principaux qui sont également ceux que nous allons utiliser dans la suite de ce manuscrit.

— Modèle de l'Oracle Aléatoire (*Random Oracle Model*, ROM) : Dans le ROM un algorithme public(souvent une fonction de hachage) peut être modélisé par un oracle aléatoire. Cela signifie qu'une personne extérieure au système ne peut pas distinguer une vraie sortie d'un algorithme d'une sortie aléatoire. Le formalisme permettant de les utiliser dans des preuves de sécurité fut présenté en 1993 par Bellare et Rogaway dans [BR93].

— Modèle Standard (*Standard Model*) : Dans ce modèle, la sécurité d'un protocole repose uniquement sur la complexité des hypothèses utilisées. On peut notamment citer la difficulté à factoriser de grands entiers en temps polynomial. Ainsi, l'adversaire est seulement limité par ses capacités calculatoires. Dans ce modèle, il est possible d'utiliser une *common reference string* (crs) *i.e.* une chaîne de caractère tirée avec une distribution donnée D. Cette chaîne de caractères est disponible pour tous les utilisateurs.

Afin de démontrer la sécurité de nos différents protocoles, nous essayons de nous ramener à un problème reconnu difficile (*e.g.* DDH ou SXDH). En partant de notre protocole de chiffrement ou de signature, nous le modifions étape par étape afin de nous rapprocher le plus possible du problème difficile. Ces modifications sont appelées couramment jeux de sécurité et furent proposées par Shoup dans [Sho02] et [Sho01]. Le i-ème jeu de sécurité est noté G_i.

Le dernier jeu permet de quantifier l'avantage de l'adversaire ou sa probabilité de succès à résoudre le problème difficile.

2.1.3 Outils Cryptographiques

Dans cette section, nous présentons différents outils mathématiques qui sont utilisés dans les protocoles cryptographiques. Ces outils permettent d'obtenir différentes propriétés suivant leurs utilisations dans les protocoles. Par exemple, nous utilisons une preuve à divulgation nulle de connaissance dans notre protocole de signature en blanc 4.2.2.

Fonctions de Hachage

Les fonctions de hachage permettent de générer une empreinte numérique d'une donnée. Comme une empreinte physique, elle se doit d'être unique. De plus, cette empreinte numérique doit permettre de reconnaître facilement la donnée associée. On note par $\mathcal{H}$ un ensemble de fonction de hachage.

Définition 17. *Une fonction de hachage H_K est une fonction de $\{0,1\}^*$ à valeur dans $\{0,1\}^k$ ou dans $\mathbb{Z}_p$.*

En cryptographie, on utilise également des fonctions de hachage universelle à sens-unique (*universal one-way function*) également appelées fonctions de hachage résistante aux collisions ciblées.

Définition 18 (*Universal one-way function*[NY89])**.** *Une famille de fonction de hachage $\mathcal{H}$ est appelée universelle à sens unique, si pour tout adversaire $\mathcal{A}$, il lui est difficile de trouver un scalaire x tel qu'il puisse trouver la seconde pré-image pour une fonction $H_K \in \mathcal{H}$.*

Les fonctions universelles à sens unique sont par exemple utilisées dans le chiffrement de Cramer-Shoup [CS98] ;

Définition 19. *Une famille $\mathcal{H}$ de fonctions de hachage H_K est résistante aux collisions si pour tout adversaire $\mathcal{A}$ il est difficile de trouver une collision sur une fonction $H_K \xleftarrow{\$} \mathcal{H}$. Nous avons :*

$$\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A}) = \Pr[H_K \xleftarrow{\$} \mathcal{H}, (m_0, m_1) \leftarrow \mathcal{A}(H_K) : H_K(m_0) = H_K(m_1)],$$

$$\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) = \max_{A \leq t} \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A}).$$

Par exemple, dans le chiffrement basé sur l'identité de Boneh-Franklin [BF01], deux fonctions de hachage sont utilisées.

Mise en gage

Cette notion est utilisée en cryptographie depuis plusieurs décennies comme par exemple par Blum [Blu81] ou Even [Eve81]. Elle fut formalisée par Brassard *et al.* dans [BCC88].

Dans une mise en gage ou *commitment*, l'utilisateur donne en garantie une partie d'une information à une personne ou à un serveur mais l'utilisateur ne lui révèle pas sa valeur. Plus tard, il a la possibilité de la dévoiler en prouvant que c'est bien celle qu'il a gagé.

D'un point de vue algorithmique, un *commitment* est composé de trois algorithmes :

— $\mathsf{Setup}(1^{\mathfrak{K}})$: génère les paramètres globaux du système.

— $\mathsf{Commit}(m; r)$: génère le *commitment* c du message m avec $r \xleftarrow{\$} \mathbb{Z}_p$.

— $\mathsf{Decommit}(c, m; w)$: ouvre c et le message m avec un témoin w qui prouve que c n'est pas corrompu.

Un *commitment* doit vérifier deux propriétés :

— La phase de création du *commit* ne doit pas laisser fuiter d'information (*hiding*).

— Lorsque que l'on retire le gage, on ne doit pas pouvoir obtenir deux valeurs différentes (*biding*).

Ces deux notions peuvent être formalisées de la manière suivante.

Définition 20 (*Hiding*)**.** *Pour un adversaire $\mathcal{A}$ il est difficile au niveau calculatoire de générer deux messages m_0 et m_1 de sorte que $\mathcal{A}$ puisse distinguer leur mise en gage c_0 et c_1. Plus formellement, pour tout adversaire $\mathcal{A}$ en temps polynomial, on a :*

$$\Pr\left[\, b = b' \;\middle|\; \begin{array}{l} \mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}}),\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{param}), b \xleftarrow{\$} \{0,1\}, \\ (c, w) \leftarrow \mathsf{Commit}(m_b) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\mathfrak{K}).$$

Définition 21 (*Binding*)**.** *Pour un adversaire $\mathcal{A}$ il est difficile au niveau calculatoire de trouver un triplet (c, d, d') tel que (c, d) et (c, d') sont des mises en gage valide pour m et m' avec $m \neq m'$. Plus formellement, pour tout adversaire $\mathcal{A}$ en temps polynomial, on a :*

$$\Pr\left[\begin{array}{l} m \neq m' \wedge \\ m \neq \perp \\ m' \neq \perp \end{array} \;\middle|\; \begin{array}{l} \mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}}),\ (c, d, d')) \leftarrow \mathcal{A}(\mathsf{param}), \\ m \leftarrow \mathsf{Decommit}(c, d), \\ m' \leftarrow \mathsf{Decommit}(c, d') \end{array}\right] \leq \mathsf{negl}(\mathfrak{K}).$$

Nous utilisons la notion de mise en gage dans la construction de notre signature en blanc 4.2.2.

Smooth Projective Hash Functions

Les *Smooth Projective Hash Functions*, notées SPHF, furent proposées par Cramer et Shoup dans [CS01]. Ces fonctions sont utilisées dans divers domaines de la cryptographie tel que l'échange de clé authentifié par mot de passe ([ACP09, GL03]), les *oblivious transfer* ([ABB+13, Kal05]) et les *oblivious signature-based envelope* ([BPV12]). Elles peuvent être vues comme un type spécifique de preuve *zero-knowledge*. Une de leur application première est la construction de protocole CCA (*i.e.* qui résiste aux attaques à messages chiffrés choisis).

Il est possible d'évaluer une fonction SPHF de deux manières différentes :

— soit en utilisant la clé (secrète) de hachage sur des mots de tout l'espace,

— soit en utilisant la clé (publique) de projection mais seulement sur un sous-ensemble de l'espace.

Les fonctions SPHF sont définies sur un Langage $\mathcal{L}$ c'est-à-dire un sous-ensemble non vide du domaine X. $\mathcal{L}$ doit être un langage $\mathcal{NP}$ *i.e.* il doit être calculatoirement difficile de distinguer si un élément aléatoire est dans $\mathcal{L}$ ou dans $X \setminus \mathcal{L}$.

Un système de *Smooth Projective Hash Function* définit sur un langage $\mathcal{L} \subset X$ sur un ensemble $\mathcal{G}$ est composée des cinq algorithmes suivants :

— $\mathsf{Setup}(\mathfrak{K})$ génère les paramètres globaux du système ainsi que la définition d'un langage $\mathcal{L}$.

— HashKG génère la clé de hachage hk.

— $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, W)$ dérive la clé de projection hp, qui peut dépendre du mot W.

— $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, W)$ renvoie la valeur hachée de W à partir de hk.

— $\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, W, w)$ renvoie la valeur hachée du mot c à partir de la clé de projection hp et du témoin w qui prouve que $W \in \mathcal{L}$.

En ce qui concerne la sécurité, un tel protocole doit vérifier les propriétés suivantes :

— *Correctness* : Soit $W \in \mathcal{L}$ et w un témoin associé. Alors, pour toute clé de hachage hk et toute clé de projection associée hp, on a :

$$\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, W) = \mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, W, w).$$

— *Smoothness* : Pour tout $W \in X \setminus \mathcal{L}$, les distributions suivantes sont statistiquement indiscernables :

$$\Delta_0 = \left\{ (W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{hk} = \mathsf{HashKG}(\mathcal{L}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, W), \\ v = \mathsf{Hash}(\mathsf{hk}, \mathcal{L}, W). \end{array} \right\}$$

$$\Delta_1 = \left\{ (W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{hk} = \mathsf{HashKG}(\mathcal{L}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, W), \\ v \xleftarrow{\$} \mathcal{G}. \end{array} \right\}.$$

Autrement dit, l'algorithme de hachage Hash est indépendant de la clé de projection hp.

— *Pseudo-randomness* : Si $W \in \mathcal{L}$, sans témoin associé, les deux distributions précédentes doivent toujours être indiscernables. Pour tout adversaire $\mathcal{A}$, la fonction d'avantage suivante est négligeable :

$$\begin{aligned} \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SPHF},\mathcal{A}}(\mathfrak{K}) = |&\Pr_{\Delta_1}[\mathcal{A}(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1] \\ &- \Pr_{\Delta_0}[\mathcal{A}(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1]| \end{aligned}$$

Un exemple d'instanciation d'une SPHF dérivant du chiffrement linéaire est présenté dans la partie 3.1.1 de ce manuscrit.

Remarque 2. *Des variantes des fonctions* SPHF *existent. On peut par exemple citées des* SPHF *avec des portes dérobées* (TSPHF) *comme présentées par Benhamouda et Pointcheval dans [BP13].*

Preuves à divulgation nulle de connaissance

Les preuves à divulgation nulle de connaissance ou *Zero-knowledge proofs* (ZK) furent introduites par Goldwasser *et al.* dans [GMR85]. De tels procédés permettent à un utilisateur de prouver la véracité d'un *statement* $\mathcal{S}$ sans donner d'autres informations. Les applications de ces preuves sont nombreuses comme par exemple pour conserver l'anonymat dans des signatures ou encore pour développer le vote électronique.

Deux parties entrent en jeu dans de tels algorithmes : le vérificateur et le *prover*.

Ces preuves doivent vérifier trois propriétés :

— *Completeness* : Si $\mathcal{S}$ est vrai, le vérificateur honnête en est convaincu.

— *Soundness* : Si $\mathcal{S}$ est faux, aucun *prover* malicieux ne peut convaincre un vérificateur honnête que $\mathcal{S}$ est vrai sauf avec une probabilité négligeable.

— *Zero-knowledge* : Tout ce qui est calculable à partir de la preuve l'est également de l'assertion.

Il est possible de supprimer l'intéraction entre le *prover* et le vérifieur. On parle alors de *non-interactive zero-knowledge proof system* (NIZK). La transformation permettant de passer d'une ZK à une NIZK a été proposée par Fiat et Shamir dans [FS87]. Il faut attendre Groth et Sahai dans [GS08] pour obtenir une méthodologie efficace pour prouver les équations de couplage dans le modèle standard. Cependant, il y a eu des travaux intermédiaires comme [BFM90] ou [DDO+01] par exemple.

Définition 22 (Non-Interactive Zero Knowledge Argument)**.** *Soit une distribution aléatoire* $\mathcal{D}_{\mathsf{param}}$ *(pour un ensemble de paramètres* param *publique) et un ensemble de langages* $\{\mathcal{L}\}_\rho$ *pour une valeur aléatoire* $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$ *et associé à une relation* $\mathcal{R}_\rho$ i.e. *un mot* x *est dans* $\mathcal{L}_\rho$ *si et seulement s'il existe un témoin* w *tel que* $\mathcal{R}(x; w)$ *existe. Un argument* NIZK*, noté* Π*, pour le langage* $\{\mathcal{L}\}_\rho$ *est composé de cinq algorithmes ppt :* $\Pi(\mathsf{Setup}, \mathsf{Gen}_{\mathsf{crs}}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Simulate})$ *:*

— $\mathsf{Setup}(\mathfrak{K})$: *renvoie les paramètres* param *du système.*

— $\mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}; \rho)$: *retourne une* crs *et une* trapdoor tk.

— $\mathsf{Prove}(\mathsf{crs}, x, w)$: *si* $(x, w) \notin \mathcal{R}_\rho$*, renvoie erreur, sinon renvoie une preuve* π.

— $\mathsf{Verify}(\mathsf{crs}, x, \pi)$: *renvoie 1 si la preuve* π *est correcte, 0 sinon.*

— $\mathsf{Simulate}(\mathsf{crs}, \mathsf{tk}, x)$: *génère une preuve* π *pour le mot* x.

Ces algorithmes doivent vérifier les propriétés suivantes :

— *Perfect completeness* : pour tout adversaire $\mathcal{A}$, tout paramètre de sécurité $\mathfrak{K}$, tout $\mathsf{param} \leftarrow \mathsf{Setup}(\mathfrak{K})$, tout nombre aléatoire $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$ et tout $(x, w) \in \mathcal{R}_\rho$, l'égalité suivante est vérifiée :

$$\Pr[(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}; \rho); \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) : \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1] = 1.$$

— *Perfect soundness* : Pour $\mathfrak{K}$ donné, un schéma vérifie la propriété de *soundness* si pour tout adversaire $\mathcal{A}$, on a :

$$\Pr[\mathsf{param} \leftarrow \mathsf{Setup}(\mathfrak{K}); \rho \leftarrow \mathcal{D}_{\mathsf{param}}; (\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho); (x, \pi) \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{crs}; \rho) : x \notin \mathcal{L}_\rho \wedge \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1] = 0.$$

— *Perfect zero-knowledge* : pour tout $\mathfrak{K}$, tout $\mathsf{param} \leftarrow \mathsf{Setup}(\mathfrak{K})$, tout $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$, tout $(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}; \rho)$ et tout couple $(x, w) \in \mathcal{R}_\rho$, les distributions $\mathsf{Prove}(\mathsf{crs}, x, w)$ et $\mathsf{Simulate}(\mathsf{crs}, \mathsf{tk}, x)$ sont identiques.

Code d'authentification de message (MAC)

Les codes d'authentification de message sont, par leur fonction, proche des signatures électroniques. Ils permettent d'assurer l'intégrité des données qu'ils accompagnent tout en permettant de vérifier si ces données ont subi des modifications durant leur transmission. Toutefois, à la différence des signatures, les MAC font partie du domaine de la cryptographie à clé privée ou symétrique.

Un MAC est composé de trois algorithmes $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{Tag}, \mathsf{Ver})$ décrit comme suit :

— $\mathsf{Gen}_{\mathsf{MAC}}$: génère la clé secrète $\mathsf{sk}_{\mathsf{MAC}}$.

— $\mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, m)$: génère un tag τ pour un message $m \in \mathcal{M}$.

— $\mathsf{Ver}(\mathsf{sk}_{\mathsf{MAC}}, m, \tau)$: renvoie 1 si le tag est correct, 0 sinon.

Un MAC est correct si l'algorithme Ver renvoie 1 pour un tag τ. Concernant la sécurité, un MAC doit être sémantiquement sûr.

Dans différentes constructions, nous utilisons des MAC dits affines présentées dans la section 3.1.2 de ce chapitre. Afin de prouver la sécurité de notre protocole de chiffrement, nous utilisons des MAC affines qui reposent sur les *Smooth Projective Hash Functions*. Nous les présentons à la suite des MAC affines.

2.2 Assurer la confidentialité : le Chiffrement

Comme nous avons pu le voir dans l'introduction, la protection des données est un enjeu depuis plusieurs milliers d'années. Il est donc naturel de présenter un des fondements de la cryptographie : le chiffrement.

Il est important de noter qu'ici nous présentons seulement la définition associée à la cryptographie à clé publique et non pas celle associée à la clé privée.

Un protocole de chiffrement $\mathcal{E}$ est composé de quatre algorithmes :

— $\mathsf{Setup}(1^{\mathfrak{K}})$: génère les paramètres param du système en utilisant le paramètre de sécurité.

— $\mathsf{KeyGen}(\mathsf{param})$: génère une paire de clé $(\mathsf{ek}, \mathsf{dk})$ avec ek la clé (publique) de chiffrement et dk la clé (secrète) de déchiffrement.

— $\mathsf{Enc}(\mathsf{ek}, m; \mu)$: génère le chiffré c du message m avec la clé ek avec la valeur aléatoire μ.

— $\mathsf{Decrypt}(\mathsf{dk}, c)$: déchiffre c afin de retrouver le message m ou $\perp$.

Concernant la sécurité, un protocole de chiffrement doit vérifier les propriétés de sécurité suivantes :

— *Correctness* : Pour toute paire de clés $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}$, tout message $m \in \mathcal{M}$, et pour tout $\mu \xleftarrow{\$} \mathbb{Z}_p$, nous avons $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, m; \mu)) = 1$.

— Sécurité Sémantique (*Indistinguishability under chosen plaintext attack*)[GM82] : un adversaire ne peut pas savoir pour quel message le chiffré qu'il possède a été chiffré. La figure 2.1 présente cette propriété sous forme de jeux. Elle est abrégée IND-CPA.

$\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ek})$
4. $c^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(c^*)$
6. Retourne b'

FIGURE 2.1 – Jeu de sécurité pour l'IND-CPA.

— *Indistinguishability under chosen ciphertext attack* [NY90, RS92] : cette propriété de sécurité est plus forte que la précédente. Par conséquent, elle est plus difficile à atteindre. Il existe deux cas différents de cette propriété :

  — le cas non-adaptatif (IND-CCA1) présenté par [NY90], l'adversaire $\mathcal{A}$ ne peut effectuer des requêtes à l'oracle de déchiffrement seulement avant l'accès au challenge.

  — le cas adaptatif (IND-CCA2) présenté par [RS92], l'adversaire peut continuer à utiliser l'oracle de déchiffrement sauf sur le chiffré challenge.

  Un adversaire ne peut pas savoir quel message a été chiffré même s'il choisit les deux messages et qu'il demande des déchiffrements pour des chiffrés qui sont différents du challenge. On la note IND-CCA2.

$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ek}, \mathsf{ODecrypt}(.))$
4. $c^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(c^*, \mathsf{ODecrypt}(.))$
6. Si $c^* \in \mathsf{CT}$ renvoie 0
7. Sinon retourne b'

FIGURE 2.2 – Jeu de sécurité pour l'IND-CCA2.

Dans la figure, l'algorithme $\mathsf{ODecrypt}()$ renvoie le déchiffrement de c en utilisant la clé de déchiffrement challenge dk. Après le déchiffrement, c est ajouté à la liste des CT des chiffrés déchiffrés.

Remarque 3. *La propriété* IND-CCA2 *est difficile à atteindre pour un protocole de chiffrement. Plusieurs propositions existent en utilisant des preuves à divulgation nulle de connaissance toutefois, même si le protocole est résistant aux attaques adaptatives à chiffrés choisis, il est peu efficace.*

Un protocole de chiffrement peut être proposé avec une propriété supplémentaire : un label. La seule différence avec un protocole de chiffrement classique est la présence d'un label publique ℓ dans l'algorithme de chiffrement. Le protocole de chiffrement est alors appelé chiffrement étiqueté (*labeled encryption*) On en rappelle la définition formelle ainsi qu'une propriété de sécurité associée.

— $\mathsf{Setup}(1^{\mathfrak{K}})$: génère les paramètres globaux du système param.

— $\mathsf{KeyGen}(\mathsf{param})$: génère une paire de clé $(\mathsf{ek}, \mathsf{dk})$ avec ek la clé (publique) de chiffrement et dk la clé (secrète) de déchiffrement.

— $\mathsf{Enc}^{\ell}(\ell, \mathsf{ek}, m; \mu)$: génère le chiffré c du message m avec la clé ek et le label ℓ avec probabilité μ.

— $\mathsf{Decrypt}^{\ell}(\mathsf{dk}, c)$: déchiffre c afin de retrouver le message m ou $\perp$.

Un tel protocole correct si pour toute paire de clé $(\mathsf{ek}, \mathsf{dk})$, tout label ℓ, pour tout $\mu \xleftarrow{\$} \mathbb{Z}_p$ et pour tout message m :

$$\mathsf{Decrypt}^{\ell}(\mathsf{dk}, \mathsf{Enc}^{\ell}(\ell, \mathsf{ek}, m; \mu)) = m.$$

De plus, un protocole étiqueté vérifie une nouvelle propriété de sécurité la *Plaintext-Checking Attack* proposé par [OP01] et elle est notée IND-PCA.

$\mathsf{Exp}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
0. $\mathsf{CT} \leftarrow [\,]$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\ell^*, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk})$
4. $c^* \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(c^*, \texttt{Oracle}(.))$
6. Si $(\ell^*, c^*) \in \mathsf{CT}$ retourne 0
5. Sinon retourne b'

FIGURE 2.3 – Jeu de sécurité pour l'IND-PCA.

Nous rappelons que $*$ en exposant signifie qu'il s'agit du chiffré ou du label challenge.

L'oracle $\texttt{Oracle}(.)$ présent à l'étape 5 prend en entrée (ℓ, c, m) et retourne une réponse seulement si le résultat du déchiffrement de c sous le label ℓ est le message m. Ensuite, il ajoute (ℓ, c) à la liste CT.

Nous avons vu trois propriétés de sécurité différentes. Il est possible d'établir des relations entre elles comme le montre le schéma suivant.

Un tel protocole est une base de la cryptographie qui a permis au fil des décennies de construire des variantes adaptées à différentes applications. Par exemple, on peut remplacer la clé

FIGURE 2.4 – Relation entre différentes propriétés de sécurité.

publique par l'identité du destinataire du message ou encore permettre seulement à une personne vérifiant certains critères de pouvoir le déchiffrer. Dans ce manuscrit nous allons notamment étudier le chiffrement de l'identité.

2.2.1 Chiffrement basé sur l'identité

Cette section porte sur une des primitives utilisées régulièrement dans ce manuscrit : le chiffrement basé sur l'identité. L'idée d'utiliser l'identité afin de chiffrer un secret fut énoncer pour la première fois par Shamir en 1984 dans [Sha84]. Toutefois, il faut attendre 2001 avant que les deux premiers protocoles soient présentés. Le premier, proposé par Boneh-Franklin [BF03] repose sur le couplage de Weil. Le second, proposé par Cocks [Coc01a] utilise les résidus quadratiques. L'IBE proposé par Boneh-Francklin est sémantiquement sûr dans le modèle de l'oracle aléatoire. Le premier IBE prouvé sémantiquement sûr dans le modèle standard est celui de Boneh-Boyen dans [BB04].

Un chiffrement basé sur l'identité, *Identity-based Encryption*, utilise l'identité du destinataire afin de chiffrer le message qui lui est destiné. Une analogie possible est celle de la boîte aux lettres. Toute personne connaissant notre adresse peut nous adresser un courrier mais nous sommes les seuls à pouvoir ouvrir cette boîte à l'aide de notre clé personnelle.

D'un point de vue algorithmique, un IBE est composé de quatre algorithmes tout comme un chiffrement à clé publique classique. Nous pouvons les définir de la manière suivante :

- $\mathsf{Setup}(1^{\mathfrak{K}})$ génère à l'aide du paramètre de sécurité les paramètres généraux du système param. Il génère également la clé publique maîtresse mpk et la clé privée maîtresse msk.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ génère la clé personnelle de l'utilisateur associée à son identité $\mathsf{usk}[\mathsf{id}]$ à l'aide de msk.
- $\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{id}, m)$ génère le chiffré c du message m en utilisant l'identité du destinataire et la mpk.
- $\mathsf{Decrypt}(\mathsf{usk}[\mathsf{id}], c)$ permet à l'utilisateur d'obtenir le message à partir du chiffré à l'aide de sa clé personnelle.

Les clés maîtresses sont utilisées par l'autorité de confiance, *trust authority*, afin de générer les clés pour chaque utilisateur.

Concernant la sécurité, un chiffrement basé sur l'identité peut vérifier deux critères : l'anonymat et la sécurité sémantique.

Anonymat.

La notion d'anonymat pour un IBE fut proposé par Abdalla *et al.* dans [ABC+05]. Un chiffrement basé sur l'identité est anonyme si un adversaire ne peut pas savoir pour quelle identité le message qu'il a reçu a été chiffré. De même que pour la sécurité sémantique, le challenger commence par générer deux chiffrés : le premier sera généré pour une identité id_0 et le deuxième pour id_1. L'adversaire devra alors déterminer pour quelle identité le chiffré qu'il a reçu a été généré. S'il réussit, il a gagné le jeu. On peut schématiser cette propriété de sécurité de la manière suivante (figure 2.5) :

En générale, on note l'anonymat d'un chiffrement basé sur l'identité par ANON-ID-CPA.

Sécurité Sémantique.

Un chiffrement est sémantiquement sûr si l'adversaire ne peut pas distinguer quel message a été chiffré entre deux messages donnés. Le challenger va chiffrer deux éléments différents : le premier sera un chiffré aléatoire, le deuxième est obtenu honnêtement en utilisant l'algorithme Encrypt. L'adversaire doit alors déterminer s'il a reçu le vrai chiffré ou un chiffré aléatoire. S'il réussit, il gagne le jeu de sécurité et le protocole n'est pas sémantiquement sûr. Le schéma suivant, figure 2.6, synthétise cette propriété de sécurité.

De manière générale, on note la sécurité sémantique d'un chiffrement utilisant l'identité par IND-ID-CPA.

Initialize :	$\underline{\mathsf{Enc}(\mathsf{id}^*, m^*)}$:
$(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$	//une seule requête
$b \xleftarrow{\$} \{0,1\}$	$\mathsf{K}_0^* \xleftarrow{\$} \mathcal{K}$
Return mpk	$(\mathsf{K}_1^*, c^*) \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*)$
	Return (K_b^*, c^*)
$\underline{\mathsf{USKGenO}(\mathsf{id})}$:	
$\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}\}$	$\underline{\mathsf{Finalize}(b')}$:
$\mathsf{usk}[\mathsf{id}] \xleftarrow{\$} \mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$	Return $(\mathsf{id}^* \notin \mathsf{ID}) \wedge (b = b')$
Return usk[id]	

FIGURE 2.5 – Schéma de sécurité pour l'anonymat.

Initialize :	$\underline{\mathsf{Enc}(\mathsf{id}^*, m^*)}$:
$(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$	//une seule requête
$b \xleftarrow{\$} \{0,1\}$	$c_0^* \xleftarrow{\$} \mathcal{C}$
Return mpk	$c_1^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m^*)$
	Return c_b^*
$\underline{\mathsf{USKGenO}(\mathsf{id})}$:	
$\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}\}$	$\underline{\mathsf{Finalize}(b')}$:
$\mathsf{usk}[\mathsf{id}] \xleftarrow{\$} \mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$	Return $(\mathsf{id}^* \notin \mathsf{ID}) \wedge (b = b')$
Return usk[id]	

FIGURE 2.6 – Schéma de sécurité pour la sécurité sémantique.

Tout comme il existe des variantes de protocole de chiffrement classique, on peut modifier les IBE afin de leur rajouter des propriétés. Il existe notamment les :

— *Blind* IBE. Notion présentée par [GH07]. Lorsqu'un utilisateur demande une clé à l'autorité, cette dernière ne sait pas pour quelle identité elle la génère.

— *Hierarchical* IBE. Notion présentée par Horwitz et Lynn dans [HL02], proposée dans le modèle de l'oracle aléatoire par Gentry et Silverberg dans [GS02] et proposée par Boneh et Boyen sans utiliser d'oracles aléatoires dans [BB04]. Ils permettent notamment de chiffrer un message à partir d'une série de bits ordonnés (*hierarchical*) et de sa clé privée. Une instanciation est proposée dans la section 4.1. Cette dernière nous permet de construire une signature à base d'attributs.

Un IBE peut être transformer en *Identity-Based Encapsulation Key Mechanism* noté IBKEM. Dans un IBKEM, l'expéditeur ne choisit pas un message à envoyer. L'algorithme de chiffrement, appelé *Encap*, génère un chiffré ainsi qu'une clé. Le destinataire devra alors retrouver la clé en utilisant le chiffré. Il est possible de transformer un IBKEM en IBE en utilisant un chiffrement symétrique (*one-time secure symmetric cipher*).

Définition 23. *Un* IBKEM *est composé de quatre algorithmes :*

— $\mathsf{Setup}(1^{\mathfrak{K}})$ *génère à l'aide du paramètre de sécurité les paramètres généraux du système* param*. Il génère également la clé publique maîtresse* mpk *et la clé privée maîtresse* msk*.*

— $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ *génère la clé de l'utilisateur associée à son identité* $\mathsf{usk}[\mathsf{id}]$ *à l'aide de* msk*.*

— $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$ *génère le chiffré* c *ainsi que la clé* K *en utilisant l'identité du destinataire et la* mpk*.*

— $\mathsf{Decrypt}(\mathsf{usk}[\mathsf{id}], c)$ *permet à l'utilisateur de retrouver la clé* K *à partir du chiffré à l'aide de sa clé personnelle.*

On note par $\mathcal{K}$ l'ensemble des clés K.

Concernant la sécurité d'un IBKEM, ce dernier vérifie les mêmes hypothèses qu'un IBE *i.e.* sécurité sémantique et anonymat.

Nous allons maintenant présenter différentes primitives de chiffrement utilisées au fil de ce manuscrit.

2.2.2 Downgradable IBE

Comme on vient de le voir il existe des variantes au chiffrement classique. Ces dernières ont été proposées au fil des années suivant l'évolution de la technologie et surtout de l'utilisation de celle-ci dans la vie courante. Par exemple, le chiffrement de l'identité permet une meilleure gestion des clés et facilite leur génération. Cette nouvelle primitive peut également permettre l'obtention d'un ABE comme nous allons le voir dans la suite.

Le *Downgradable* IBE fut introduit dans [BGP19]. Son principe est de pouvoir dériver la clé de l'utilisateur pour en obtenir des nouvelles. Cette dérivation gagne en efficacité, notamment en termes de génération d'éléments aléatoires.

Définition 24. *Un* Downgradable IBE *est composé de cinq algorithmes :* $(\mathsf{Setup}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{USKDown})$. *Les quatre premiers algorithmes sont ceux d'un* IBE *classique. L'algorithme* USKDown *est défini comme suit :*

$\mathsf{USKDown}(\mathsf{usk}[\mathsf{id}], \tilde{\mathsf{id}})$ *: renvoie la clé secrète de l'utilisateur* $\mathsf{usk}[\tilde{\mathsf{id}}]$ *tant que* $\tilde{\mathsf{id}} \preceq \mathsf{id}$.

Tout comme un IBE, on souhaite qu'un DIBE soit anonyme et sémantiquement sûr. Nous allons voir dans la suite de ce manuscrit, que l'instanciation que nous utilisons vérifie ces deux hypothèses grâce à l'IBE sous-jacent. Une construction générale est présentée dans la section 3.1.4.

Transformation d'un DIBE en ABE

Nous utilisons cette transformation afin d'obtenir une signature à partir d'attributs. Avant de présenter cette construction, montrons comment à partir d'un ensemble d'attributs S nous obtenons une identité.

Pour tout sous-ensemble $S \subseteq \mathbb{U}$, on définit $\mathsf{id}_S \in \{0,1\}^\ell$ où le i-ème bit est défini par :

$$\mathsf{id}_S[i] := \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{sinon} \end{cases}$$

Ainsi, à partir d'un ensemble d'attributs S, nous obtenons une identité qui est associée à S.

$\underline{\mathsf{Setup}(\mathsf{param})}$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathsf{DIBE}}(1^{\mathfrak{K}})$
Return $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{USKGen}(\mathsf{sk}, \mathbb{A})}$:
$\mathsf{usk}[\mathbb{A}] \leftarrow \mathsf{USKGen}_{\mathsf{DIBE}}(\mathsf{sk}, \mathbb{A})$
Return $\mathsf{usk}[\mathbb{A}]$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mathbb{F}, m)}$
$\mathbb{F} = \bigvee_{j=1}^{k} (\bigwedge_{a \in S_j} a)$
$\forall j \in [A, k]$, calcule :
$(c_j, K_j) \leftarrow \mathsf{Enc}_{\mathsf{DIBE}}(\mathsf{pk}, \mathsf{id}_{S_j})$ et $c'_j \leftarrow m \oplus K_j$
Return $c = (c_1, ..., c_k, c'_1, ..., c'_k)$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, c)}$:
$\mathbb{F} = \bigvee_{j=1}^{k} (\bigwedge_{a \in S_j} a)$
Cherche $j \in [1, k]$ tel que $S_j \subseteq \mathbb{A}$
Calcule : $U \leftarrow \mathsf{USKDown}_{\mathsf{DIBE}}(\mathsf{usk}[\mathbb{A}], \mathsf{id}_{S_j})$
Calcule $K_j \leftarrow \mathsf{Dec}_{\mathsf{DIBE}}(U, \mathsf{id}_{S_j}, c_j)$
Return $M = c'_j \oplus K_j$

FIGURE 2.7 – ABE à partir d'un DIBE.

La construction précédente est utilisée afin de construire un chiffrement basé sur les attributs. Cette transformation est utilisée dans notre signature 4.1.

2.2.3 Chiffrement par attributs

Comme dit précédemment, nous pouvons utiliser l'identité d'une personne afin de chiffrer un message qu'il lui est destiné. Cette modification d'un protocole de chiffrement fait partie d'une famille plus large de schéma : le chiffrement par attributs (*Attribute-based encryption* ABE). En effet, au lieu de chiffrer pour des utilisateurs vérifiant différents attributs on ne chiffre que pour un seul utilisateur. Ce chiffrement fut proposé par Sahai et Waters dans [SW05].

Définition 25 (Access Structure)**.** *Une structure d'accès est une collection non-vide de sous-ensemble de l'univers des attributs* $\mathbb{U}$*.*

Deux versions des chiffrements par attributs existent : une qui repose sur la politique d'accès sur la clé (KP-ABE) et l'autre qui repose sur la politique d'accès sur le message chiffré (CP-ABE). Ces deux versions furent proposées par Goyal *et al.* dans [GPSW06]. Dans la suite, nous allons nous concentrer sur les politiques d'accès sur le chiffré. On note par $\mathbb{A}$ l'ensemble des attributs de l'utilisateur.

Définition 26 (CP-ABE)**.** *Un schéma de chiffrement par attributs est composé de quatre algorithmes :*

- $\mathsf{Setup}(1^{\mathfrak{K}})$ *: génère une paire de clés maîtresses* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{USKGen}(\mathsf{sk}, \mathbb{A})$ *: retourne la clé secrète de l'utilisateur* $\mathsf{usk}[\mathbb{A}]$ *associée à un ensemble d'attributs* $\mathbb{A} \cup \mathbb{U}$
- $\mathsf{Enc}(\mathsf{pk}, m, \mathbb{F})$ *: renvoie le chiffré* c *en accord avec la structure d'accès* $\mathbb{F}$*.*
- $\mathsf{Dec}(\mathsf{usk}[\mathbb{A}], c, \mathbb{A}, \mathbb{F})$ *: retourne le message* m *ou* $\perp$*.*

En ce qui concerne la *perfect correctness*, il faut que pour tout $\mathfrak{K} \in \mathbb{N}$, tout couple $(\mathsf{pk}, \mathsf{sk})$, toute structure d'accès $\mathbb{F}$, tout ensemble d'attributs $\mathbb{A} \subset \mathbb{U}$, toute clé $\mathsf{usk}[\mathbb{A}]$ et tout chiffré c généré par $\mathsf{Enc}(\mathsf{pk}, m, \mathbb{F})$:

$$\Pr[\mathsf{Dec}(\mathsf{usk}[\mathbb{A}], c, \mathbb{A}, \mathbb{F}) = m] = 1.$$

Dans la suite nous étudions les signatures issues de tels chiffrements (cf. 2.3.3) que nous utilisons dans une de nos constructions de signature électronique (cf. 4.1).

2.3 Assurer l'intégrité : Les Signatures Électroniques

Les signatures électroniques sont une analogie aux signatures manuscrites. En cryptographie, elles permettent d'assurer l'intégrité du message et d'authentifier le signataire. Cette nouvelle primitive cryptographique fut proposée par Diffie et Hellman dans leur article fondateur [DH76]. La première instantiation sûre fut présentée par Goldwasser, Micali et Rivest dans [GMR88].

Un protocole de signature électronique est composé des quatre algorithmes suivants :

- $\mathsf{Setup}(1^{\mathfrak{K}})$: génère les paramètres globaux du système param.
- $\mathsf{KeyGen}(\mathsf{param})$: génère la paire de clé $(\mathsf{sk}, \mathsf{vk})$ avec sk la clé (secrète) de signature et vk la clé (publique) de vérification.
- $\mathsf{Sign}(\mathsf{sk}, m; \mu)$ génère la signature σ du message m à l'aide de la clé sk et d'une valeur aléatoire μ.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: vérifie la signature σ à l'aide de la clé vk.

Concernant la sécurité, une signature doit vérifier les propriétés suivantes :

— *Correctness* : Pour toute paire de clés $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, pour tout message $m \in \mathcal{M}$, et pour tout $\mu \xleftarrow{\$} \mathbb{Z}_p$, on a : $\mathsf{Verify}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m; \mu)) = 1$. Dans ce cas, on dit que la signature σ est valide.

— *Existential Unforgeability under Chosen Message Attacks* [GMR88]. Autrement dit un adversaire ne doit pas pouvoir générer une signature valide après avoir obtenu n signatures valides. Le jeu de sécurité associé à cette propriété est présenté dans la figure 2.8 :

$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \mathsf{OSign}(\mathsf{vk}, \cdot))$
4. $b \leftarrow \mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*)$
5. Si $m^* \in \mathcal{SM}$ renvoie 0
6. Sinon renvoie b

FIGURE 2.8 – Jeu de sécurité pour l'$\mathsf{EUF} - \mathsf{CMA}$.

Le protocole énoncé plus haut est une signature classique sans autre propriété. De même que pour les protocoles de chiffrement on peut construire des signatures qui seront plus adaptées à certaines utilisations. Par exemple, il existe des signatures en blanc ou encore des signatures qui permettent de signer sur des chiffrés en étant "randomisables". Les définitions de ces différentes signatures sont présentées dans la suite de cette section.

2.3.1 Signature en blanc

Les signatures en blanc, les *blind signatures*, furent introduites par Chaum dans [Cha82]. L'utilisation première de ces signatures est la volonté de ne pas pouvoir tracer les paiements. Le serveur, noté $\mathcal{S}$, qui signe le message dans un tel protocole n'apprend aucune information sur ce message. D'une certaine manière, le serveur signe en "aveugle" sans pouvoir lire ce qu'il signe. Une autre utilisation de ces signatures peut être la gestion de monnaies électroniques dont on a pu voir une certaine expansion ces dernières années.

Définition 27 (*Blind Signature*)**.** *Un protocole de signature en blanc, noté $\mathcal{BS}$, est composé de trois algorithmes et d'un protocole interactif :*

— $\mathsf{BSSetup}(1^\lambda)$ *: génère les paramètres* param *du système.*

— $\mathsf{BSKeyGen}(\mathsf{param})$ *: génère la clé de signature et la clé de vérification :* $(\mathsf{sk}, \mathsf{vk})$*.*

— $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}, (\mathsf{vk}, m)\rangle$ *: est un protocole interactif entre l'utilisateur $\mathcal{U}$ et le signataire/serveur $\mathcal{S}$. Il retourne une signature σ sur le message m.*

— $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ *: vérfie si σ est une signature valide. Il retourne 1 si σ est valide, 0 sinon.*

Les deux propriétés de sécurité attendues sont la résistance aux contrefaçons, qui protège le signataire, et la *blindness*, qui protège l'utilisateur. Les figures 2.9 et 2.10 présentent ces propriétés sous forme de jeux de sécurité.

Une construction générique des signatures en blanc fut proposée par Fischlin dans [Fis06] dans le modèle *common reference string*. Les auteurs de [AFG+10] proposent le premier schéma concret utilisant cette construction. Cependant, ils n'utilisent pas une hypothèse de sécurité standard. Dans la section 4.2.2, nous utilisons cette construction afin d'élaborer une signature en blanc en un tour dans le modèle standard [BBCF20].

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\mathfrak{K})$
1. $(\mathsf{param}) \leftarrow \mathsf{BSSetup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{BSKeyGen}(\mathsf{param})$
3. Pour $i = 1, \ldots, q_s$, $\mathsf{BSProtocol}\langle \mathcal{S}(\mathsf{sk}), \mathcal{A}(\mathsf{vk})\rangle$
4. $\big((m_1, \sigma_1), \ldots, (m_{q_s+1}, \sigma_{q_s+1})\big) \leftarrow \mathcal{A}(\mathsf{vk})$;
5. Si $\exists i \neq j, m_i = m_j$ ou si $\exists i, \mathsf{Verify}(\mathsf{vk}, m_i, \sigma_i) = 0$
renvoie 0
6. Sinon renvoie 1

FIGURE 2.9 – Résistance aux contrefaçons pour un schéma de $\mathcal{BS}$.

$\mathsf{Exp}^{\mathsf{bl}-b}_{\mathcal{BS},\mathcal{S}^*}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{BSSetup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk}, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{param})$
3. $\sigma_b \leftarrow \mathsf{BSProtocol}\langle \mathcal{A}, \mathcal{U}(\mathsf{vk}, m_b)\rangle$
4. $\sigma_{1-b} \leftarrow \mathsf{BSProtocol}\langle \mathcal{A}, \mathcal{U}(\mathsf{vk}, m_{1-b})\rangle$
5. $b^* \leftarrow \mathcal{S}^*(m_0, m_1)$;
6. Retourne $b^* = b$.

FIGURE 2.10 – *Blindness* pour un schéma de $\mathcal{BS}$.

2.3.2 Transformation de Fischlin

Afin de construire notre signature en blanc (section 4.2.2), nous utilisons une construction de Marc Fischlin [Fis06] qui permet d'effectuer des signatures en blanc de manière optimale.

Le protocole de signature est interactif entre l'utilisateur et l'organisme qui doit signer (par exemple une banque).

Cette signature utilise une preuve à divulgation nulle de connaissance non-interactive (NIZK). Nous notons par :

— $\mathcal{P}$ le *prover*

— $\mathcal{V}$ la personne qui vérifie la signature

— $\mathsf{Setup}_{\mathsf{ZK}}$ l'algorithme associée à la preuve à divulgation nulle de connaissance

— Setup_{com} l'algorithme associée à la mise en gage

— $\mathsf{USKGen}_{\mathsf{Enc}}$ l'algorithme qui génère les clés de chiffrement/déchiffrement.

Nous rappelons la définition de la transformation de Fischlin dans la figure 2.11.

Cette construction peut être représentée sous forme schématique afin de mettre en valeur l'unique échange entre l'utilisateur et le signataire comme le montre la figure 2.12.

2.3.3 Signature par attributs

Les signatures utilisant des attributs ou *attribute-based signature*, notées ABS, furent introduites par Maji *et al.* dans [MPR11]. Notons par $\mathbb{U}$ l'univers des attributs. Dans une telle signature, le signataire qui possède un ensemble d'attributs, noté $\mathbb{A}$, peut signer un message si ses attributs vérifient le prédicat Υ associé à la signature. Avant d'introduire ces signatures, présentons différentes notions afin de pouvoir définir un prédicat Υ.

Définition 28 (Span Program [KW93])**.** *Soit $\mathbb{K}$ un corps commutatif fixé. Un* span program *sur $\mathbb{K}$ est une matrice $\hat{M}(M, \rho)$ où M est une matrice sur $\mathbb{K}$ et $\rho : rows(M) :\rightarrow \{x_i^\epsilon | i \in [n], \epsilon = 0, 1\}$.*

$\underline{\mathsf{Setup}(1^{\mathfrak{K}})}$:
$\mathsf{crs}_{\mathsf{ZK}} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^{\mathfrak{K}})$
$\mathsf{crs}_{com} \leftarrow \mathsf{Setup}_{com}(1^{\mathfrak{K}})$
$(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{USKGen}_{\mathsf{Enc}}(1^{\mathfrak{K}})$
Return $\mathsf{crs}_{\mathcal{BS}} = (\mathsf{crs}_{\mathsf{ZK}}, \mathsf{crs}_{com}, \mathsf{ek})$

$\underline{\mathsf{USKGen}(\mathsf{param})}$:
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{USKGen}_{\mathcal{BS}}$
Return $(\mathsf{sk}, \mathsf{vk})$

$\underline{\mathsf{Sign}(\mathsf{ek}, m)}$
$u, v \xleftarrow{\$} \{0,1\}^n$
$U \leftarrow \mathsf{Commit}(\mathsf{crs}, m; u)$
$B \leftarrow \mathsf{Sign}(\mathsf{sk}, U)$
Si $\mathsf{Verify}(\mathsf{vk}, U, B) \neq 1$, renvoie $\perp$.
$x \leftarrow C||\mathsf{pk}||\mathsf{crs}_{com}||\mathsf{vk}||m$
$w \leftarrow u||v||B$
$\pi \leftarrow \mathcal{P}(\mathsf{crs}_{\mathsf{ZK}}, c, w, \mathsf{sk})$

$\underline{\mathsf{Verify}(\mathsf{crs}_{\mathcal{BS}}, S, m)}$:
$S = c||\pi$
Return le résultat obtenu par $\mathcal{V}(\mathsf{crs}_{\mathsf{ZK}}, x, \pi)$

FIGURE 2.11 – Signature en blanc proposée par Fischlin.

Utilisateur		Signataire
$U \leftarrow \mathsf{Commit}(\mathsf{crs}_{\mathsf{Com}}, m; u)$		
	$\xrightarrow{\quad U \quad}$	
		$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, U; \tau)$
	$\xleftarrow{\quad \sigma_{\mathsf{SRC}} \quad}$	
Si $\mathsf{Verify}_{\mathsf{SRC}}(\mathsf{vk}, U, \sigma) = 1$ alors :		
$C = \mathsf{Encrypt}(\mathsf{pk}, U, B)$		
$\pi = \mathsf{NIZK.Prove}(C$ est une signature valide de m)		

FIGURE 2.12 – Vue simplifiée de la construction de Fischlin.

Un *span program* est dit monotone si l'image de ρ est seulement des littéraux positifs (ou atomes) de la forme $\{x_1, ..., x_n\}$.

Maintenant, on peut définir ce qu'est un prédicat Υ. On considère une fonction booléenne $\Upsilon : \{0,1\}^n \to \{0,1\}$. Un *span program* monotone pour Υ sur un corps $\mathbb{F}$ est une matrice M de taille $\ell \times t$ associée à un label $a : [\ell] \to [n]$ qui associe à chaque colonne de M une variable que prend en entrée Υ. Ainsi, pour chaque $\{x_1, ..., x_n\} \in \{0,1\}^n$, on a :

$$\Upsilon(x_1, ..., x_n) = 1 \Leftrightarrow \exists v \in \mathbb{F}^{1\times\ell} | vM = [1, 0, 0, ..., 0] \text{ et } \forall i : x_{a(i)} = 0 \Rightarrow v_i = 0.$$

Définition 29. *On dit qu'un ensemble d'attributs $\mathbb{A}$ satisfait un prédicat Υ si $\Upsilon(\mathbb{A}) = 1$.*

Définition 30 (ABS)**.** *Un schéma de signature basé sur des attributs est composé de quatre algorithmes :*

- $\mathsf{Setup}(1^{\mathfrak{K}})$: *génère les paramètres du système ainsi qu'une paire de clés* $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbb{A})$: *génère la clé de signature de l'utilisateur à partir de ses attributs* $\mathsf{usk}_{\mathbb{A}}$.
- $\mathsf{Sign}(\mathsf{usk}_{\mathbb{A}}, m)$ *: signe le message m à l'aide* $\mathsf{usk}_{\mathbb{A}}$ *où* $\Upsilon(\mathbb{A}) = 1$. *Renvoie la signature σ sur m.*
- $\mathsf{Verify}(m, \sigma, \mathsf{mpk})$: *renvoie 1 si σ est valide, 0 sinon.*

Concernant la sécurité, ces signatures doivent respecter la *perfect privacy i.e.* la vie privée du signataire ne dépend que de la signature et non de l'autorité.

Un schéma ABS respecte cette propriété si l'égalité suivante est vérifiée :

$$\mathsf{Sign}(\mathsf{param}, \mathsf{sk}_1, m, \Upsilon) = \mathsf{Sign}(\mathsf{param}, \mathsf{sk}_2, m, \Upsilon)$$

avec : $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{param}, \mathbb{A}_i)$ et $\Upsilon(\mathbb{A}_1) = \Upsilon(\mathbb{A}_2) = 1$.

Bien entendu, comme tout protocole de signature, les signatures par attributs doivent vérifier la *correctness* et résister aux contrefaçons.

2.3.4 *Designated Verifier Signature*

Une signature qui permet de désigner un vérificateur, notée DVS donne la possibilité au signataire de désigner une personne qui sera le seul à pouvoir la vérifier. Il est possible d'étendre cette notion à plusieurs vérificateurs. Nous en proposons d'ailleurs une à la section 4.1. Le schéma présenté dans la définition suivante fut proposé par Steinfeld *et.al* dans [SBWP03].

Définition 31. *Un schéma de signature dont on peut désigner un vérifieur est composé de sept algorithmes :*

- $\mathsf{Setup}(1^{\mathfrak{K}})$: *génère les paramètres globaux du système* param.
- $\mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param})$: *génère une paire de clés pour le signataire* $(\mathsf{sk}_1, \mathsf{pk}_1)$.
- $\mathsf{KeyGen}_{\mathcal{V}}(\mathsf{param})$: *génère une paire de clés de vérification pour le vérificateur* $(\mathsf{sk}_2, \mathsf{pk}_2)$.
- $\mathsf{Sign}(\mathsf{sk}_1, m)$ *génère la signature* σ *sur le message* m *vérifiable avec la clé* pk_1.
- $\mathsf{Verify}(\mathsf{pk}_1, m, \sigma)$: *vérifie la validité de la signature* σ.
- $\mathsf{Des}(\mathsf{pk}_1, \mathsf{pk}_2, m, \sigma)$: *génère une signature* $\hat{\sigma}$ *pour le vérifieur. Il pourra vérifier cette signature à l'aide de sa clé secrète.*
- $\mathsf{DesV}(\mathsf{sk}_2, \mathsf{pk}_1, m, \hat{\sigma})$ *vérifie la validité de* $\hat{\sigma}$.

Concernant la sécurité, cette signature doit vérifier, comme un protocole de signature classique la résistance aux contrefaçons. En accord avec sa propriété, elle doit vérifier la DV-*unforgeability* ainsi que la non-transférabilité qui sont présentées dans la suite.

- DV-*unforgeability* : Seul le signataire ou le vérificateur peut générer une signature vérifiable $\hat{\sigma}$ pour un message m. Cette propriété est présentée sous forme de jeu dans la figure 2.13.

$\mathsf{Exp}^{\mathsf{UDVS}}_{\mathsf{dvuf},\mathcal{A}}(\mathfrak{K})$
1. $(\mathsf{param}) \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param})$
3. $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{KeyGen}_{\mathcal{V}}(\mathsf{param})$
4. $(m, \hat{\sigma}) \leftarrow \mathcal{A}(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{OSign}, \mathsf{OVerif})$
6. `RETURN` $\mathsf{DesV}(\mathsf{sk}_2, \mathsf{pk}_1, m, \hat{\sigma})$ if $m \notin \mathcal{SM}$

FIGURE 2.13 – DV-*unforgeability*.

Deux oracles sont utilisés dans la figure précédente :

- $\mathsf{OSign}(\mathsf{pk}_1, m)$: génère une signature σ sur le message m et ajoute ce message à l'ensemble des messages signés $\mathcal{SM}$.
- $\mathsf{OVerif}(\mathsf{pk}_2, m, \hat{\sigma})$: vérifie la validité de la signature $\hat{\sigma}$.

- Non-transférabilité : Un adversaire ne peut pas convaincre une personne extérieure au système de la validité (ou de l'invalidité) d'une signature $\hat{\sigma}$. L'avantage de l'adversaire de distinguer les deux distributions suivantes doit être négligeable :

$$\Delta_0 = \left\{ (m, \hat{\sigma}) \;\middle|\; \begin{array}{l} (\mathsf{sk}_1, \mathsf{pk}_1), (\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{KeyGen}_{\mathcal{S}}, \mathsf{KeyGen}_{\mathcal{V}} \\ \hat{\sigma} = \mathsf{Des}(\mathsf{pk}_1, \mathsf{pk}_2, m, \mathsf{Sign}(\mathsf{sk}_1, m)) \end{array} \right\}$$

$$\Delta_1 = \left\{ (m, \hat{\sigma}) \;\middle|\; \begin{array}{l} (\mathsf{sk}_1, \mathsf{pk}_1), (\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{KeyGen}_{\mathcal{S}}, \mathsf{KeyGen}_{\mathcal{V}} \\ \hat{\sigma} \leftarrow \mathcal{S} \end{array} \right\}.$$

2.3.5 Signature sur un chiffré *randomizable*

Les signatures sur un chiffré *randomizable* furent introduites par Blazy *et al.* dans [BFPV11]. L'idée est de combiner les chiffrements *randomizables* avec un protocole de signature. Ces protocoles permettent de transformer un chiffré déjà existant en un nouveau chiffré mais sur le même message. L'utilisation principale de cette signature est la construction d'une signature en blanc en tour optimal.

Définition 32. *Un schéma de signature sur un chiffré pouvant être randomisé* (SRC) *est composé des sept algorithmes suivants :*

- $\mathsf{Setup}(1^{\mathfrak{K}})$ *: génère les paramètres globaux* param *pour les schémas de chiffrement et de signature.*
- $\mathsf{KeyGen}_{\mathcal{E}}(\mathsf{param})$ *: génère une paire de clé de chiffrement/déchiffrement* $(\mathsf{ek}, \mathsf{dk})$.
- $\mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param})$ *: génère une paire de clé de vérification/singnature* $(\mathsf{vk}, \mathsf{sk})$.
- $\mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}, m; r)$ *: renvoie le chiffré c du message $m \in \mathcal{M}$ sous la clé* ek *en utilisant le jeton aléatoire $r \in \mathcal{R}$.*
- $\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; s)$ *: à l'aide de $s \in \mathcal{R}$, renvoie la signature σ de c ou $\perp$ si c n'est pas valide*
- $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{vk}, c)$ *: déchiffre c en utilisant la clé* dk*. Il renvoie le message m ou $\perp$.*
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, c, \sigma)$ *: renvoie 1 si σ est valide, 0 sinon.*
- $\mathsf{Random}(\mathsf{vk}, \mathsf{ek}, c, \sigma; r')$ *: renvoie un chiffré c' associé au même message m et une signature σ' de c'.*

On dit qu'une signature sur un chiffré est *ciphertext randomizable* si la signature randomisée sur un chiffré randomisé est statistiquement indiscernable d'une nouvelle signature.

Une des instanciations proposées dans [BFPV11] utilise la signature de Waters présentée dans la partie suivante.

2.3.6 Signature préservant la structure

Les *Structure Preserving Signature*, notée SPS, sont des signatures dont tout les éléments (signature, message, clés, etc ...) sont des éléments de groupes. Cette nouvelle primitive fut proposée par Abe *et al.* dans [AFG+10].

Définition 33 (*Structure Preserving Signature*)**.** *Une signature préservant la structure est composée de trois algorithmes :*

- $\mathsf{Gen}(1^{\mathfrak{K}})$ *: génère les paramètres du système ainsi qu'une paire de clés publique/privée* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, \mathbf{m})$ *: génère la signature σ sur le message $\boldsymbol{m}$ à l'aide de la clé* sk.
- $\mathsf{Verify}(\sigma, \mathsf{pk}, \mathbf{m})$ *: renvoie 1 si la signature est valide et 0 sinon.*

Les définitions précédentes sont le socle de la cryptographie moderne. Les différents outils sont utilisés afin de proposer de nouveaux protocoles plus adaptés aux enjeux du monde moderne. Les définitions formelles de chiffrement et de signature permettent de comprendre plus facilement les prochaines constructions.

Le chapitre suivant introduit plusieurs constructions de chiffrement et de signature utilisées dans la suite de cette thèse. Ces constructions sont présentées avec leurs propriétés de sécurité.

Chapitre 3

Constructions Génériques

Lors de la présentation de la cryptographie à clé publique par Diffie et Hellman dans *New directions in cryptography* [DH76], les auteurs présentent une nouvelle méthode afin de faciliter l'échange de clé entre différents utilisateurs. Rapidement après, un protocole pratique est présenté, le protocole RSA [RSA78]. Toutefois, ce ne fut pas toujours le cas comme par exemple pour les chiffrements reposant sur l'identité. L'idée d'utiliser de tels chiffrements fut proposés par Shamir en 1984 [Sha84] toutefois ce n'est qu'en 2001 que les deux premiers protocoles pratiques furent proposés ([BF01, Coc01b]). Ainsi, en cryptographie, les réponses ne sont pas toujours immédiates et il faut attendre l'avancée de la technologie ou des théories mathématiques pour permettre de proposer des réponses concrètes.

Le chapitre précédent était consacré à la présentation des concepts de la cryptographie à clé publique tels que le chiffrement et les signatures électroniques. Dans ce chapitre, nous allons présenter les solutions pratiques *i.e.* les instanciations. Sont notamment présentés : le protocole de chiffrement ElGamal[ElG85] et l'IBE de Boneh-Franklin [BF01] pour les plus connus. À l'aide de ces derniers, nous pourrons par la suite, proposer de nouveaux protocoles dont certains répondent à des problèmes ouverts.

Les constructions énoncées dans la suite sont présentées avec leur propriété de sécurité.

Sommaire

3.1 Constructions Génériques de chiffrements

Comme on a pu le voir, le chiffrement est une des primitives les plus connue de la cryptographie. Il permet d'assurer la confidentialité du message échangé entre plusieurs entités.

Dans cette section, différents protocoles de chiffrement sont présentés. Les premiers, dits classiques, sont des bases qui ont permis de proposer des chiffrements plus élaborés ou qui sont prouvés sûr dans des modèles de sécurité différents.

Des chiffrements plus spécifiques, comme par exemple le *tight* IBE de Blazy *et al.* [BKP14] est utilisé dans plusieurs de nos constructions.

3.1.1 Chiffrements classiques

Chiffrement à clé publique de ElGamal

Ce chiffrement fut proposé par ElGamal en 1984 dans [ElG84]. Comme les différents chiffrements présentés et étudiés dans la suite de ce manuscrit, il fait partie des chiffrements à clé publique.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}})}$:	$\underline{\mathsf{Enc}(\mathsf{ek}, m)}$
Return $(\mathbb{G}, p, [1])$.	$r \xleftarrow{\$} \mathbb{Z}_p$
	Return $c = (c_1, c_2) = ([r, rh + m])$.
$\underline{\mathsf{USKGen}(\mathsf{param})}$:	
$h \xleftarrow{\$} \mathbb{Z}_p$	$\underline{\mathsf{Decrypt}(\mathsf{dk}, c)}$:
Return $(\mathsf{ek}, \mathsf{dk}) = ([h], h)$.	Return $[m] = [c_2 - \mathsf{dk}c_1]$.

FIGURE 3.1 – Chiffrement de ElGamal.

Ce schéma est sémantiquement sûr (IND-CPA) sous l'hypothèse DDH.

Ce chiffrement est utilisé dans notre signature en blanc présentée dans le chapitre 5.3, section 4.2.2.

Chiffrement de Cramer-Shoup

Le chiffrement de Cramer-Shoup [CS98] est une généralisation du chiffrement de ElGamal à la sécurité IND-CCA2. Historiquement, ce chiffrement est le premier à vérifier les trois propriétés les plus attendues d'un chiffrement : il est résistant aux attaques adaptatives à chiffrés choisis, cette propriété est prouvée sûre dans le modèle standard et il est efficace.

On note par $\mathcal{H}$ l'ensemble des fonctions de hachage résistantes aux collisions.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}})}$:	$\underline{\mathsf{Enc}(\ell, \mathsf{ek}, m; r)}$:
Return param.	$r \xleftarrow{\$} \mathbb{Z}_p$
	$e = [m + hr]$
$\underline{\mathsf{KeyGen}(\mathsf{param})}$:	$\xi = H_K(r, e)$
$\mathsf{dk} = (x_1, x_2, y_1, y_2, z_1, z_2) \xleftarrow{\$} \mathbb{Z}_p^5$	$C = (r, e, v = (c + d\xi)r)$
Posons : $c = [x_1 + x_2], d = [y_1 + y_2]$	Return C
et $h = [z_1 + z_2]$	
$H_K \leftarrow \mathcal{H}$	$\underline{\mathsf{Dec}(\ell, \mathsf{dk}, C)}$:
$\mathsf{ek} = (c, d, H_K)$	Calcule $\xi = H_K(r, e)$ et vérifie quand :
Return dk, ek	$rx_1 + r\xi y_1 + rx_2 + r\xi y_2 \stackrel{?}{=} v$
	Calcule $m = [e - r(z_1 + z_2)]$ ou renvoie $\perp$.

FIGURE 3.2 – Chiffrement de Cramer-Shoup.

Concernant la sécurité, ce protocole est IND-CCA2 sous les hypothèses DDH et de résistance aux collisions pour la fonction de hachage utilisée.

Une version labellisée de ce protocole est présentée dans le dernier chapitre de ce manuscrit (6.4).

Chiffrement Linéaire

Le chiffrement linéaire ou *linear encryption* fut introduit par Boneh, Boyen et Sacham dans [BBS04].

Cette construction est sémantiquement sûre sous l'hypothèse DLin.

$\underline{\mathsf{Setup}(1^\kappa)}$:
$g \leftarrow \mathbb{G}$
Return $\mathsf{param}(\mathbb{G}, p, g)$

$\underline{\mathsf{USKGen}(\mathsf{param})}$:
$\mathsf{dk} = (y_1, y_2) \xleftarrow{\$} \mathbb{Z}_p^2$
$\mathsf{ek} = (\mathsf{ek}_1, \mathsf{ek}_2) = ([y_1], [y_2])$
Return $(\mathsf{ek}, \mathsf{dk})$

$\underline{\mathsf{Enc}(\mathsf{ek}, m)}$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
Return $c = (c_1, c_2, c_3) = ([y_1 r_1], [y_2 r_2], [r_1 + r_2 + m])$.

$\underline{\mathsf{Decrypt}(\mathsf{dk}, c)}$:
Return $[m] = [c_3 - (c_1/y_1 + c_2/y_2)]$.

FIGURE 3.3 – Chiffrement Linéaire.

Une *smooth projective hash function* peut dériver de ce chiffrement comme le montre l'exemple suivant.

Dans cet exemple, nous allons considérer le langage $\mathcal{L} = \mathsf{Lin}(\mathsf{ek}, m)$ *i.e.* le chiffrement linéaire de m sous la clé ek. Dans ce cas, le chiffré de m est noté c.

On peut maintenant présenter une SPHF associé au chiffrement linéaire en reprenant les notations précédentes :

— $\mathsf{HashKG}(\mathcal{L})$ retourne la clé de hachage $\mathsf{hk} = (x_1, x_2, x_3) \xleftarrow{\$} \mathbb{Z}_p^3$.
— $\mathsf{ProjKG}(\mathsf{hk}; \mathcal{L})$ retourne la clé de projection $\mathsf{hp} = ([y_1 x_1 + x_3], [y_2 x_2 + x_3])$.
— $\mathsf{Hash}(\mathsf{hk}, c; \mathcal{L})$: retourne le haché suivant : $[y_1 r_1 x_1 + y_2 r_2 x_2 + (r_1 + r_2) x_3]$.
— $\mathsf{ProjHash}(\mathsf{hp}, c, r_1, r_2)$ retourne le résultat de $[y_1 x_1 r_1 + x_3 r_1 + y_2 x_2 r_2 + x_3 r_2]$.

Afin de continuer l'exemple, nous allons montrer que cette SPHF est correcte, *smooth* et *pseudo-random*.

Théorème 1. *La* smooth projective hash function *présentée plus haut est correcte.*

Démonstration. Montrons que les algorithmes Hash et ProjHash renvoient le même résultat.

$$\mathsf{Hash}(\mathsf{hp}, \mathcal{L}, c) = [y_1 r_1 x_1 + y_2 r_2 x_2 + (r_1 + r_2) x_3]$$
$$\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, c, r_1, r_2) = [y_1 x_1 r_1 + x_3 r_1 + y_2 x_2 r_2 + x_3 r_2]$$

Ainsi, les deux équations sont égales. On peut donc en conclure que notre SPHF est correcte. □

Théorème 2. *La* smooth projective hash function *présentée plus haut est* smooth.

Démonstration. Pour montrer que la SPHF est *smooth*, nous allons montrer que si le haché est un élément aléatoire, la distribution est indiscernable d'une distribution classique. Pour cela, nous allons montrer que le haché, noté v est indépendant de c et de hp. v est donc de la forme $v = \mathsf{Hash}(\mathsf{hk}, \mathcal{L}, c)$.

Supposons que hp soit connue et que c est sous la forme d'un chiffré incorrect : $c = (\mathsf{ek}_1^{r_1}, \mathsf{ek}_2^{r_2}, [r_3] \cdot m)$ avec $r_3 \neq r_1 + r_2$.

$$\begin{pmatrix} \log \mathsf{hp}_1 \\ \log \mathsf{hp}_2 \\ \log v \end{pmatrix} = \underbrace{\begin{pmatrix} y_1 & 0 & 1 \\ 0 & y_2 & 1 \\ y_1 r_1 & y_2 r_2 & r_3 \end{pmatrix}}_{M} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Le déterminant de la matrice M est $y_1 y_2 (r_3 - r_1 - r_2)$. Le déterminant est non-nul si c n'appartient pas au langage (*i.e.* $r_3 \neq r_1 + r_2$). On en conclut donc que v est indépendant de hp et de c. □

Théorème 3. *La* smooth projective hash function *présentée plus haut est* pseudo-random *sous l'hypothèse* DLin.

Démonstration. Le chiffrement linéaire est sémantiquement sûre sous l'hypothèse DLin. Nous allons montrer que la *pseudo-randomness* de la SPHF est vérifiée grâce à cette propriété du chiffrement.

Considérons deux messages $m \neq m'$. Posons $\mathcal{E}_{\mathsf{Lin}}$ la fonction de chiffrement du chiffrement linéaire utilisant la clé publique associée ek. Ainsi, $\mathcal{E}_{\mathsf{Lin}}(m)$ et $\mathcal{E}_{\mathsf{Lin}}(m')$ sont indiscernables sous l'hypothèse DLin.

De plus, le résultat précédent permet de savoir que les distributions

$$\mathcal{D}_0 = \{\mathsf{Lin}(\mathsf{ek}, m), c = \mathcal{E}_{\mathsf{Lin}}(m), v \xleftarrow{\$} \mathcal{G}\} \text{ et } \mathcal{D}_1 = \{\mathsf{Lin}(\mathsf{ek}, m), c = \mathcal{E}_{\mathsf{Lin}}(m), v = \mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, m), c)\}$$

sont indiscernables.

Si l'on considère les quatre distributions suivantes :

$$\mathcal{D}_0 = \{\mathsf{Lin}(\mathsf{ek}, m), c = \mathcal{E}_{\mathsf{Lin}}(m), v \xleftarrow{\$} \mathcal{G}\},$$
$$\mathcal{D}_1 = \{\mathsf{Lin}(\mathsf{ek}, m), c = \mathcal{E}_{\mathsf{Lin}}(m), v = \mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, m), c)\},$$
$$\mathcal{D}_2 = \{\mathsf{Lin}(\mathsf{ek}, m), c = \mathcal{E}_{\mathsf{Lin}}(m'), v \xleftarrow{\$} \mathcal{G}\},$$
$$\mathcal{D}_3 = \{\mathsf{Lin}(\mathsf{ek}, m), c = \mathcal{E}_{\mathsf{Lin}}(m'), v = \mathsf{Hash}(\mathsf{hk}, \mathsf{Lin}(\mathsf{ek}, m), c)\},$$

en utilisant la sécurité sémantique du chiffrement, on sait que $\mathcal{D}_0$ et $\mathcal{D}_2$ et $\mathcal{D}_1$ et $\mathcal{D}_3$ sont indistinguables. On en déduit que $\mathcal{D}_2$ et $\mathcal{D}_3$ sont indiscernables. □

IBE de Boneh-Franklin

Le premier chiffrement basé sur l'identité utilisant le couplage de Weil fut présenté en 2001 par Boneh et Franklin. La figure 3.4 présente la construction. On considère l'espace des messages $\mathcal{M} = \{0,1\}$ pour $n \in \mathbb{N}$.

$\underline{\mathsf{Setup}(1^\mathfrak{K})}$:
$\mathsf{param} \xleftarrow{\$} \mathsf{GGen}(1^\mathfrak{K})$
$H : \{0,1\}^* \to \mathbb{G}$
$H_T : \{0,1\}^* \to \{0,1\}^n$
$s \xleftarrow{\$} \mathbb{Z}_p^*$, $h := [s]$
Return $\mathsf{msk} = s$
$\mathsf{mpk} = (\mathbb{G}, \mathbb{G}_T, H, H_T, e, h)$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})}$:
Return $\mathsf{usk}[\mathsf{id}] = [sH(\mathsf{id})]$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)}$
Return $c = (c_1, c_2) = ([r], m \oplus H_T([srH(\mathsf{id})]_T))$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, c)}$:
Return $m = c_2 \oplus H_T([\mathsf{usk}[\mathsf{id}]c_1]_T)$

FIGURE 3.4 – IBE de Boneh-Franklin.

Ce schéma est sémantiquement sûr sous l'hypothèse BDH dans le modèle de l'oracle aléatoire.

3.1.2 Tight IBE

Les protocoles présentés dans la suite du manuscrit sont basés sur un IBE présenté par Blazy *et al.* dans [BKP14]. Ce protocole est dit *tight*, car il permet d'avoir un coût minimal de réduction. Dans d'autres constructions, pour obtenir un tel résultat il est nécessaire de réduire la taille des paramètres. Ainsi, nous en rappelons la construction dans la suite. Nous allons commencer par présenter les MAC affines qui sont utilisés dans ce protocole.

Notions nécessaires

Dans la première partie des définitions (2.1.3), nous avons présenté les MAC classiques. Toutefois, cette construction nécessite l'utilisation de MAC dit affines dont nous rappelons la définition.

Définition 34. *Soit $\mathfrak{K}$ les paramètres du système contenant un groupe $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q)$ d'ordre premier p et soit $n, n' \in \mathbb{N}$. On dit qu'un* $\mathsf{MAC} = (\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{Tag}, \mathsf{Ver})$ *est affine sur $\mathbb{Z}_p^{n \times n'}$ si les conditions suivantes sont vérifiées :*

1. $\mathsf{Gen}_{\mathsf{MAC}}(\mathfrak{K})$ *retourne* $\mathsf{sk}_{\mathsf{MAC}}$ *contenant $(\boldsymbol{B}, \mathbf{x}_0, \ldots, \mathbf{x}_\ell, x')$, où $\boldsymbol{B} \in \mathbb{Z}_p^{n \times n'}$, $\mathbf{x}_i \in \mathbb{Z}_p^n$, $x' \in \mathbb{Z}_p$, pour un $n', l \in \mathbb{N}$.*
2. $\mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, \mathbf{m} \in \mathcal{B}^l)$ *retourne un tag $\tau = ([\boldsymbol{t}]_2, [u]_2) \in \mathbb{G}_2^n \times \mathbb{G}_2$, pour une fonction publique $f_i : \mathcal{M} \to \mathbb{Z}_p$:*

$$\boldsymbol{t} = \mathbf{B}\boldsymbol{s} \in \mathbb{Z}_p^n \quad \text{avec } \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^{n'}, \quad u = \sum_{i=0}^{l} f_i(\boldsymbol{m}) \mathbf{x}_i^\top \boldsymbol{t} + x' \in \mathbb{Z}_p$$

3. $\mathsf{Ver}(\mathsf{sk}_{\mathsf{MAC}}, \mathbf{m}, \tau = ([\mathbf{t}]_2, [u]_2))$ *vérifie si les équations précédentes sont vérifiées.*

Un MAC affine doit vérifier la *pseudorandomness against chosen message attacks* (PR-CMA). Le jeu de sécurité associé à cette propriété de sécurité est présenté dans la figure 3.5.

Initialize :	Chal($\mathbf{m}^*$) : // une requête
$\mathsf{sk}_{\mathsf{MAC}} \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(\mathsf{param})$	$h \leftarrow \mathbb{Z}_p^*$
Return $\mathsf{sk}_{\mathsf{MAC}}$	$\mathbf{h}_0 = (\sum f(\mathsf{m}_j^*) \cdot \mathbf{x}_j)h \in \mathbb{Z}_q^{k+1}$;
	$h_1 = \sum f'(\mathsf{m}_j^*) \cdot x'_j)h \in \mathbb{Z}_q$
Eval($\mathbf{m}$) :	$\boxed{\mathbf{h}_0 \xleftarrow{\$} \mathbb{Z}_q^{k+1} ; h_1 \xleftarrow{\$} \mathbb{Z}_q}$
$\mathcal{Q}_{\mathcal{M}} = \mathcal{Q}_{\mathcal{M}} \cup \{\mathsf{m}\}$	Return $([h]_1, [\mathbf{h}_0]_1, [h_1]_T)$
$([\mathbf{t}]_2, [u]_2) \leftarrow \mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, \mathbf{m})$	
Return $([\mathbf{t}]_2, [u]_2)$	Finalize($b \in \{0,1\}$) :
	Return $b \wedge (\mathsf{m}^* \notin \mathcal{Q}_{\mathcal{M}})$

FIGURE 3.5 – Jeu $\mathsf{PR\text{-}CMA}_{\mathsf{real}}$ et $\boxed{\mathsf{PR\text{-}CMA}_{\mathsf{rand}}}$.

Définition 35. *Un* MAC *affine définie sur $\mathbb{Z}_p^n$ est* PR-CMA*-secure si pour tout adversaire $\mathcal{A}$ son avantage défini comme suit est négligeable :* $\mathsf{Adv}_{\mathsf{MAC}}^{\mathsf{PR\text{-}CMA}}(\mathcal{A}) := \Pr[\mathsf{PR\text{-}CMA}_{\mathsf{real}}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{PR\text{-}CMA}_{\mathsf{rand}}^{\mathcal{A}} \Rightarrow 1]$.

Dans l'article original, les auteurs présentent cette seule notion de sécurité pour les MAC affines. Toutefois, pour la transformation que nous proposons dans la section 5.2.2. Nous avons besoin d'une notion légèrement différente que nous appelons RPR-CMA. Tout comme la sécurité PR-CMA nous la présentons sous forme d'un jeu réel (real) et d'un jeu aléatoire (rand).

Introduisons, comme la sécurité PR-CMA, la sécurité RPR-CMA sous forme de jeu réel et aléatoire.

Dans la figure 3.6, les demandes effectuées à PEval ajoutent le tag associé aux demandes dans les challenges interdits *i.e.* $\mathcal{A}$ ne pourra pas l'utiliser lors du challenge. Ce point est différent de la figure 3.7. Cependant un adversaire effectuant des requêtes à Eval ainsi qu'à PEval ne doit pas obtenir suffisamment d'information sur x'.

Cependant, ces MAC ne sont pas suffisants pour montrer que notre construction présentée dans [BBP19] est anonyme. Pour cela, nous avons besoin de MAC affines possédant un tag partiel que nous définissons dans le point suivant :

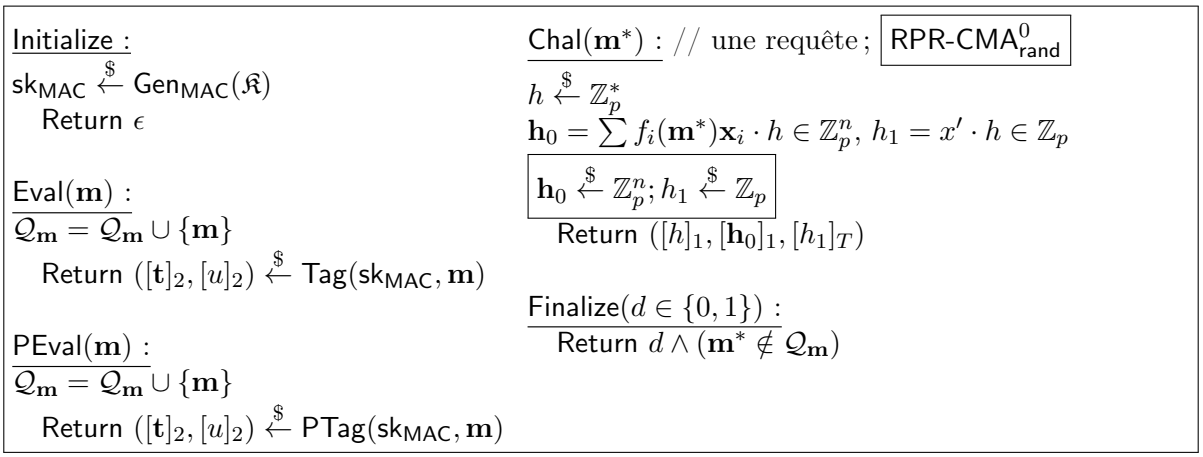


FIGURE 3.6 – Jeu RPR-CMA$^0_{\mathsf{real}}$ et RPR-CMA$^0_{\mathsf{rand}}$ pour définir la sécurité RPR-CMA^0.

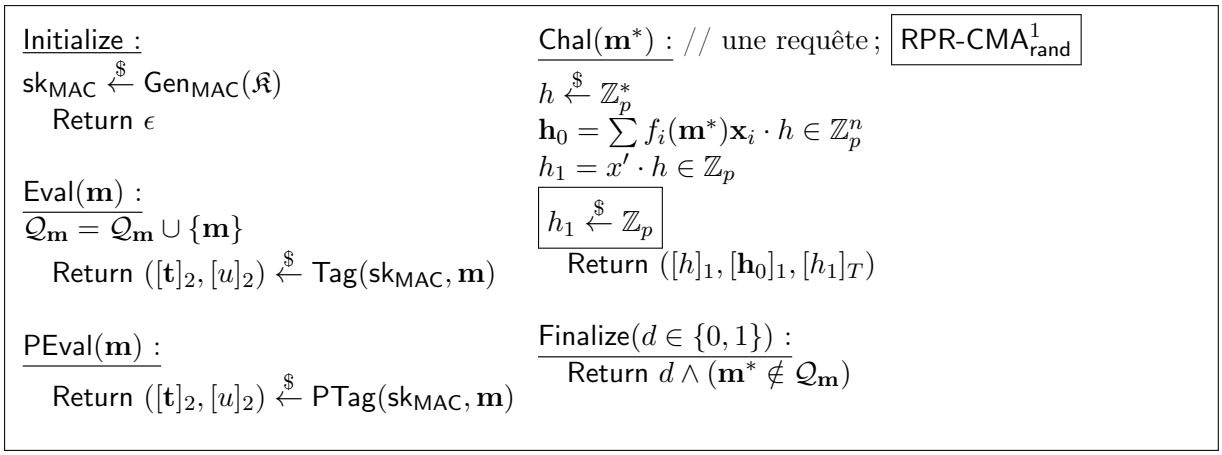


FIGURE 3.7 – Jeu RPR-CMA$^1_{\mathsf{real}}$ et RPR-CMA$^1_{\mathsf{rand}}$ pour définir la sécurité RPR-CMA^1.

$\mathsf{PTag}(\mathsf{sk}_{\mathsf{MAC}}, \mathbf{m} \in \mathcal{B}^l)$ renvoie un tag partiel $\tau = ([\mathbf{t}]_2, [u]_2) \in \mathbb{G}_2^n \times \mathbb{G}_2$, obtenu par :

$$\mathbf{t} = \mathbf{B}\mathbf{s} \in \mathbb{Z}_p^n \quad \text{avec } \mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^{n'}, \quad u = \sum_{i=0}^{l} f_i(\mathbf{m})\mathbf{x}_i^\top \mathbf{t} \in \mathbb{Z}_p.$$

Construction

La figure 3.8 présente le *tight* IBE de Blazy *et al.*. Cette construction repose sur une hypothèse classique et utilise un groupe d'ordre premier ce qui est un avantage pour l'utiliser dans diverses constructions. Nous allons notamment utiliser cette construction pour proposer une nouvelle primitive de chiffrement (5.2.2) ou de signature (4.1). On note par f une fonction définie publiquement tel que $f : \mathcal{M} \to \mathbb{Z}_p$.

Théorème 4. *Sous les hypothèses* $k -$ MDDH*, cette construction est* PR-ID-CPA*.*

3.1.3 IBE de Boyen-Waters

De même que pour la construction précédente, l'IBE de Boyen-Waters [BW06] est adapté dans la suite avec un algorithme qui permet de connaître l'identité du destinataire (cf. 5.2.1).

$\underline{\mathsf{Gen}(\mathfrak{K})}$:
$\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$
$\mathsf{sk}_{\mathsf{MAC}} = (\mathbf{B}, \mathbf{x}_0, \ldots, \mathbf{x}_\ell, x') \xleftarrow{\$} \mathsf{Gen}_{\mathsf{MAC}}(\mathfrak{K})$
$\forall i \in [\![0, \ell]\!] : \mathbf{Y}_i \xleftarrow{\$} \mathbb{Z}_p^{k\times n}$ $\mathbf{Z}_i = (\mathbf{Y}_i^\top | \mathbf{x}_i) \cdot \mathbf{A} \in \mathbb{Z}_p^{n\times k}$
$\mathbf{y'} \xleftarrow{\$} \mathbb{Z}_p^k$; $\mathbf{z'} = (\mathbf{y'}^\top | x') \cdot \mathbf{A} \in \mathbb{Z}_p^{1\times k}$
$\mathsf{mpk} = (\mathcal{G}, [\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{[\![0,\ell]\!]}, [\mathbf{z'}]_1)$,
$\mathsf{msk} = (\mathsf{sk}_{\mathsf{MAC}}, (\mathbf{Y}_i)_{[\![0,\ell]\!]}, \mathbf{y'})$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{mpk}, \mathsf{id})}$:
$([\mathbf{t}]_2, [u]_2) \xleftarrow{\$} \mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, \mathsf{id})$
$\mathbf{v} = \sum_{i=0}^{\ell} f_i(\mathsf{id})\mathbf{Y}_i\mathbf{t} + \mathbf{y'} \in \mathbb{Z}_p^k$
Return $\mathsf{usk}[\mathsf{id}] = ([\mathbf{t}]_2, [u]_2, [\mathbf{v}]_2) \in \mathbb{G}_2^{n+1+k}$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})}$:
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$
$\mathbf{c}_0 = \mathbf{Ar} \in \mathbb{Z}_p^{k+1}$, $\mathbf{c}_1 = (\sum_{i=0}^{l} f_i(\mathsf{id})\mathbf{Z}_i) \cdot \mathbf{r} \in \mathbb{Z}_p^n$
$K = \mathbf{z'} \cdot \mathbf{r} \in \mathbb{Z}_p$
Return $\mathsf{K} = [K]_T$ et $C = ([\mathbf{c}_0]_1, [\mathbf{c}_1]_1) \in \mathbb{G}_1^{n+k+1}$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, C)}$:
$K = e\left([\mathbf{c}_0]_1, \begin{bmatrix}\mathbf{v}\\ u\end{bmatrix}_2\right) / e([\mathbf{c}_1]_1, [\mathbf{t}]_2)$
Return $K \in \mathbb{G}_T$

FIGURE 3.8 – *Tight* IBE de Blazy-Kiltz-Pan.

$\underline{\mathsf{Gen}(\mathfrak{K})}$:
$f, h, \omega, t_1, \ldots, t_4 \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{msk} = \omega, t_1, \ldots, t_4$
$\mathsf{mpk} = [\omega t_1 t_2]_T, [f], [h], [t_1], \ldots, [t_4]$
Return mpk

$\underline{\mathsf{USKGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})}$:
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{usk}[\mathsf{id}] = [r_1 t_1 t_2 + r_2 t_3 t_4, -\omega t_2 - r_1 t_2(f + h\mathsf{id}), -\omega t_1 - r_1 t_1(f + h\mathsf{id}), -r_2 t_4(f + h\mathsf{id}), -r_2 t_3(f + h\mathsf{id})]$
Return $\mathsf{usk}[\mathsf{id}] \in \mathbb{G}^5$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})}$:
$s, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{K} = [s\mathsf{mpk}]_T$
$c = [s(f + h\mathsf{id}), (s - s_1)t_1, s_1 t_2, (s - s_2)t_3, s_2 t_4]$
Return $\mathsf{K}, c \in \mathbb{G}_T \times \mathbb{G}^4$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, c)}$:
$\mathsf{K}' = [\sum_i c_i \cdot \mathsf{usk}[\mathsf{id}]_i]_T$
Return $\mathsf{K}' \in \mathbb{G}_T$

FIGURE 3.9 – IBE de Boyen-Waters.

Concernant la sécurité, cette construction est sémantiquement sûre sous l'hypothèse bilinéaire de Diffie-Hellman (BDH) et respecte la propriété de l'anonymat sous l'hypothèse linéaire décisionnelle (DLin).

3.1.4 DIBE

Les deux constructions présentes dans cette section du manuscrit furent présentées par Blazy *et al.* dans [BGP19]. Ce protocole est adapté dans la section 4.1. Il est basé sur le schéma de [BKP14] présenté précédemment.

Théorème 5. *Sous l'hypothèse* $\mathcal{D}_k$-MDDH, *la construction* DIBE *est* PR-ID-CPA. *Pour tout adversaire* $\mathcal{A}$, *il existe un adversaire* $\mathcal{B}$ *tel que* $\mathcal{T}(\mathcal{A}) \approx \mathcal{T}(\mathcal{B})$ *et* $\mathsf{Adv}_{\mathsf{DIBE},\mathcal{D}_k}(\mathcal{B})^{\mathsf{PR\text{-}ID\text{-}CPA}} \leq (\mathsf{Adv}_{\mathcal{D}_k,\mathsf{Setup}}(\mathcal{B}) + 2q_k(\mathsf{Adv}_{\mathcal{D}_k,\mathsf{Setup}}(\mathcal{B})) + 1/q)$. [1]

1. q_k est le nombre de requêtes maximal que l'adversaire peut faire à l'oracle Eval.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}})}$:
$\mathbf{B} \xleftarrow{\$} \mathcal{D}_k$
For $i \in [\![1, \ell]\!]$: $y_i \xleftarrow{\$} \mathbb{Z}_p; z_i = y_i \cdot a \in \mathbb{Z}_p$
$y' \xleftarrow{\$} \mathbb{Z}_p; z' = y'^{\top} \cdot a \in \mathbb{Z}_p$;
$\mathsf{mpk} := (\mathcal{G}, [a]_1, ([z_i]_1)_{1 \le i \le \ell}, [z']_1)$
$\mathsf{msk} := ((y_i)_{1 \le i \le \ell}), [y']_1)$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})}$:
$t \xleftarrow{\$} \mathbb{Z}_p$
$v = y' + \sum_{i=1}^{\ell} \mathsf{id}_i y_i t$

$S \xleftarrow{\$} \mathbb{Z}_p$; $T = \mathbf{B} \cdot S \in \mathbb{Z}_p$
$V = \sum_{i=0}^{\ell(\mathsf{id})} \mathsf{id}_i \mathbf{Z}_i T$
For $i, \mathsf{id}[i] = 1$:
$e_i = y_i \cdot t \in \mathbb{Z}_p$;
$E_i = \mathbf{Z}_i T \in \mathbb{Z}_p$

$\mathsf{usk} := ([t]_2, [v]_2)$
$\mathsf{udk}[\mathsf{id}] := ([T]_2, [V]_2, ([e_i]_2, [E_i]_2)_{i, \mathsf{id}[i]=1})$
Return $(\mathsf{usk}[\mathsf{id}], [e_i]_2)$

$\underline{\mathsf{Enc}(\mathsf{usk}[\mathsf{id}], \tilde{\mathsf{id}})}$:
$r \xleftarrow{\$} \mathbb{Z}_p^k$
$c_0 = \mathbf{A} r \in \mathbb{Z}_p^{k+1}$
$c_1 = (\sum_{i=0}^{\ell(\mathsf{id})} \mathsf{id} \mathbf{Z}_i) \cdot r \in \mathbb{Z}_p^n$
$K = z' \cdot r \in \mathbb{Z}_p$
Return $c = ([c_0]_1, [c_1]_1)$ et $\mathsf{sk} = [K]_T$.

$\underline{\mathsf{USKDown}(\mathsf{usk}[\mathsf{id}], \tilde{\mathsf{id}})}$:
If $\neg(\tilde{\mathsf{id}} \preceq \mathsf{id})$, renvoie $\perp$.

Set $\mathcal{I} = \{i | \tilde{\mathsf{id}}[i] = 0 \wedge \mathsf{id}[i] = 1\}$
Downgrading la clé :
$\hat{v} = v + \sum_{i \in \mathcal{I}} \mathsf{id}_{S_j, i} e_i \in \mathbb{Z}_p$
$\hat{V} = V + \sum_{i \in \mathcal{I}} \mathsf{id}_{S_j, i} E_i \in \mathbb{Z}_p$
Re-randomisation de $(\hat{v}, \hat{V})$:
$s' \leftarrow \mathbb{Z}_p; S' \leftarrow \mathbb{Z}_p$
$t' = t + Ts' \in \mathbb{Z}_p$
$T' = T \cdot S'$
$\hat{v}' = \hat{v} + \hat{V} \cdot s' \in \mathbb{Z}_p$
$V' = \hat{V} \cdot S' \in \mathbb{Z}_p$
Re-randomisation de e_i :
Pour i, $\tilde{\mathsf{id}}[i] = 1$:
$e'_i = e_i + E_i s' \in \mathbb{Z}_p$
$E'_i = E_i \cdot S' \in \mathbb{Z}_p$

$\mathsf{usk}[\mathsf{id}'] = ([t']_2, [v']_2)$
$\mathsf{udk}[\mathsf{id}'] = ([T']_2, [V']_2, [e'_i]_2, [E'_i]_2)$

Return $(\mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}'])$

$\underline{\mathsf{Verify}(\mathsf{udk}[\tilde{\mathsf{id}}], c)}$:
$\mathsf{usk}[\mathsf{id}] \stackrel{?}{=} ([t']_2, [v']_2)$.
$c \stackrel{?}{=} ([c_0]_1, [c_1]_1)$.
$\mathsf{sk} = e([c_0]_1, [v]_2) \cdot e([c_1]_1, [t]_2)^{-1}$
Return $\mathsf{sk} \in \mathbb{G}_T$

FIGURE 3.10 – DIBE basé sur MDDH.

3.2 Constructions génériques de signatures

Les signatures électroniques permettent d'assurer l'authenticité d'un message. Après avoir présenté différentes constructions de chiffrement aux propriétés utiles dans la suite de ce manuscrit, nous présentons dans cette section des instanciations de signatures utilisées dans les prochains chapitres.

3.2.1 Signature de Waters

La signature suivante fut proposée par Brent Waters dans [Wat05]. Cette signature est une adaptation du schéma IBE qu'il propose dans ce même article. Tout comme pour le protocole de chiffrement, la signature utilise ce qu'on appelle couramment la fonction de Waters notée $\mathcal{F}$.

Définition 36. *On suppose l'existence d'une liste de générateurs indépendants* $[\mathbf{u}_i] \xleftarrow{\$} \mathbb{G}$. *On définit la fonction de Waters comme suit :* $[\mathcal{F}(m)] = [\mathbf{u}_0 \sum_{i=1}^{k} \mathbf{u}_i^{m_i}]$ *avec* $m \in \{0,1\}^k$.

Définition 37. *La signature de Waters est composée de quatre algorithmes :*

- $\mathsf{Setup}(1^{\mathfrak{K}})$ *: génère les paramètres globaux du systèmes* param *ainsi que deux générateurs* $g, h \in \mathbb{G}$.
- $\mathsf{KeyGen}(\mathsf{param})$ *:* $(\mathsf{vk}, \mathsf{sk}) = ([x], [hx])$ *avec* $x \xleftarrow{\$} \mathbb{Z}_p$.
- $\mathsf{Sign}(\mathsf{sk}, m)$ *: prend* $s \xleftarrow{\$} \mathbb{Z}_p$ *et calcule* $\sigma(m) = (\sigma_1, \sigma_2) = ([hx + s\mathcal{F}(m)], [s])$.

— $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: *vérifie quand* $[\sigma_1]_T \stackrel{?}{=} [\mathcal{F}(m)\sigma_2 + h\mathsf{vk}]_T$.

Théorème 6. *Cette signature résiste aux contrefaçons sous l'hypothèse décisionnel de* BDH.

Cette signature, étant issue d'un chiffrement sur l'identité, elle est utilisée dans notre protocole à trois parties du chapitre 4 (6.7). Pour être précis, dans ce dernier protocole, nous utilisons la version asymétrique de la signature *i.e.* on travaille avec deux groupes $\mathbb{G}_1, \mathbb{G}_2$.

3.2.2 Structure Preserving Signature de Kiltz *et al.*

La signature suivante fut proposée par Kiltz *et al.* dans [KPW15]. Nous utilisons ce schéma afin de construire notre signature en blanc en tour optimal présentée à la section 4.2.2. Notons que dans notre construction, qui est basée sur l'hypothèse SXDH nous avons $k = n = 1$.

$\underline{\mathsf{Setup}(\mathfrak{K})}$:
Return (param)

$\underline{\mathsf{KeyGen}(\mathsf{param})}$:
$\mathbf{A}, \mathbf{B} \stackrel{\$}{\leftarrow} \mathcal{D}_1; \mathbf{K} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{(n+1)\times(k+1)}$
$\mathbf{K_0}, \mathbf{K_1} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{(k+1)\times(k+1)}$
$\mathbf{C} := \mathbf{KA} \in \mathbb{Z}_p^{(n+1)\times k}$
$(\mathbf{C_0}, \mathbf{C_1}) := (\mathbf{K_0 A}, \mathbf{K_1 A}) \in (\mathbb{Z}_p^{(k+1)\times k})^2$
$(\mathbf{P_0}, \mathbf{P_1}) := (\mathbf{B}^\intercal \mathbf{K_0}, \mathbf{B}^\intercal \mathbf{K_1}) \in (\mathbb{Z}_p^{k\times(k+1)})^2$
$\mathbf{sk} := (\mathbf{K}, [\mathbf{P_0}]_1, [\mathbf{P_1}]_1, [\mathbf{B}]_1)$;
$\mathbf{vk} := ([\mathbf{C_0}]_2, [\mathbf{C_1}]_2, [\mathbf{C}]_2, [\mathbf{A}]_2)$

$\underline{\mathsf{Sign}(\mathsf{sk}, \mathbf{m})}$:
$\mathbf{r} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^k; \tau \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
$\sigma_1 := [(1, \mathbf{m})\mathbf{K} + \mathbf{r}^\intercal(\mathbf{P_0} + \tau \mathbf{P_1})]_1 \in \mathbb{G}_1^{1\times(k+1))}$,
$\sigma_2 := [\mathbf{r}^\intercal \mathbf{B}^\intercal]_1 \in \mathbb{G}_1^{1\times(k+1)}$,
$\sigma_3 := [\mathbf{r}^\intercal \mathbf{B}^\intercal \tau]_1 \in \mathbb{G}_1^{1\times(k+1)}$,
$\sigma_\tau := [\tau]_2 \in \mathbb{G}_2$
Return $(\sigma_1, \sigma_2, \sigma_3, \sigma_\tau)$

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathbf{m}, \sigma)}$:
$\mathtt{Parse}\ \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$
$\mathtt{Check}\ [\sigma_1 \cdot \mathbf{A}]_T = [(1, \mathbf{m}) \cdot \mathbf{C} + \sigma_2 \cdot \mathbf{C_0} + \sigma_3 \cdot \mathbf{C_1}]_T$
et $[\sigma_2 \cdot \sigma_4]_T = [\sigma_3]_T$

FIGURE 3.11 – Schéma SPS de [KPW15].

Théorème 7. *Sous l'hypothèse* $\mathcal{D}_k - \mathsf{MDDH}$ *dans* $\mathbb{G}_1$ *et* $\mathcal{D}_k - \mathsf{KerMDDH}$ *dans* $\mathbb{G}_2$ *la signature est* unbounded CMA*-secure.*

Dans le but de prouver ce théorème, les auteurs utilisent un lemme présenté dans le même article que nous énonçons maintenant.

Lemme 1 (Computational core lemma for unbounded CMA-security)**.** *Pour tout adversaire* $\mathcal{A}$*, il existe un challenger* $\mathcal{B}$ *avec* $\mathcal{T}(\mathcal{A}) \approx \mathcal{T}(\mathcal{B})$ *et*

$$\Pr\left[\begin{array}{l|l} & \mathbf{A}, \mathbf{B} \stackrel{\$}{\leftarrow} \mathcal{D}_k; \\ & \mathbf{K_0}, \mathbf{K_1} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{k+1\times k+1} \\ \tau^* \notin \mathcal{Q}_{tag} & (\mathbf{P_0}, \mathbf{P_1}) := (\mathbf{B}^\intercal \mathbf{K_0}, \mathbf{B}^\intercal \mathbf{K_1}) \in (\mathbb{Z}_p^{k\times k+1})^2 \\ \wedge b' = b & \mathbf{vk} := ([\mathbf{P_0}]_1, [\mathbf{P_1}]_1, [\mathbf{B}]_1, \mathbf{K}_0\mathbf{A}, \mathbf{K}_1\mathbf{A}, \mathbf{A}) \\ & b \stackrel{\$}{\leftarrow} \{0,1\}; b' \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathcal{O}_b(.), \mathcal{O}^*(.)}(\mathsf{vk}) \end{array}\right]$$

$$\leq \frac{1}{2} + 2Q \cdot \mathsf{Adv}^{mddh}_{\mathcal{D}_k, \mathsf{Setup}}(\mathcal{B}) + Q/q$$

où :

— $\mathcal{O}(\tau)$ *renvoie* $([b\mu\mathbf{A}^\perp + \mathbf{r}^\top(\mathbf{P}_0 + \tau\mathbf{P}_1)]_1, [\mathbf{r}^\top\mathbf{B}]_1) \in (\mathbb{G}_1^{1\times(k+1)})^2$ *avec* $\mu \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, $\mathbf{r} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^k$ *et ajoute* τ *à* $\mathcal{Q}_{msg}$*. Ici,* $\mathbf{A}^\top$ *est un vecteur non-nul de* $\mathbb{Z}_p^{1\times(k+1)}$ *qui vérifie* $\mathbf{A}^\perp\mathbf{A} = 0$.

— $\mathcal{O}^*([\tau^*]_2)$ *renvoie* $[\mathbf{K}_0 + \tau^*\mathbf{K}_1]_2$. $\mathcal{A}$ *effectue une unique demande* τ^* *à* $\mathcal{O}^*$.

— $\mathcal{Q}$ *est le nombre de requêtes de* $\mathcal{A}$ *effectuer à* $\mathcal{O}_b$.

Ce lemme est utilisé dans la preuve de résistance aux contrefaçons de notre signature SRC (4.2.1) qui repose sur la signature SPS de Kiltz.

Chapitre 4

Protection de la vie privée en utilisant des protocoles de signatures

La volonté de protéger notre vie privée est un enjeu du XXI$^{\text{ième}}$ siècle. On entend par protection de la vie privée la protection de nos données sensibles telles que des données bancaires ou médicales. Des solutions cryptographiques existent depuis plusieurs décennies afin de favoriser cette protection. Dans ce chapitre, nous allons présenter des signatures électroniques qui permettent de protéger l'anonymat et la vie privée des utilisateurs.

Les signatures par attributs permettent, en un sens de conserver l'anonymat de l'utilisateur. Elles furent introduites par Maji *et al.* [MPR11]. Une personne signe le message en possédant les attributs en adéquation avec la politique. Ainsi, les vérifieurs savent seulement qu'un individu ou une entité avec les bons attributs a pu signer le message. Ainsi, dans la signature que nous présentons dans la suite, nous gardons cette propriété d'anonymat en permettant au signataire de désigner un groupe de vérifieurs possédant les bons attributs.

La seconde signature est une signature en blanc qui permet de protéger à la fois l'utilisateur et le signataire. Les signatures en blanc existent depuis 1982 avec leur présentation par David Chaum dans [Cha82]. Toutefois le protocole présenté dans la suite est un des premiers qui s'effectue en un seul échange entre l'utilisateur et le signataire et dont la taille de la signature est constante et l'hypothèse de sécurité est une hypothèse standard (*i.e.* SXDH). Avec ce résultat est également présenté une signature sur un chiffré *randomizable* qui permet de construire la signature en blanc. La signature sur un chiffré *randomizable* peut notamment être utilisée dans le vote électronique pour prévenir les fraudes.

Sommaire

4.1 Signature utilisant des attributs

Les signatures basées sur des attributs furent présentées par Maji *et al.* [MPR11] en 2011 dont nous rappelons la définition formelle dans la section 2.3.3.

Dans cette partie du manuscrit nous allons présenter une signature utilisant des attributs et qui permet également de désigner la ou les personnes qui peuvent vérifier la signature. Un tel protocole est appelé *Attribute-based Designated Verifier Signature*, noté ABDVS. Cette construction

utilise différentes briques de cryptographie existantes tels qu'un protocole de chiffrement basée sur l'identité ainsi qu'une signature basée également sur l'identité. Nous pouvons maintenant présenter plus précisément la construction.

Dans l'Union Européenne, les organismes (privés ou publics) qui traitent les données personnelles d'utilisateurs doivent respecter le RGPD ([VVdB17]). Le Règlement Général sur la Protection des données (RGPD) est un règlement de l'Union Européenne qui permet la protection des données d'un utilisateur notamment des données digitales. Dans ce contexte, une ABDVS est totalement adaptée à l'utilisation par le *cloud* car seule une entité avec les attributs adéquats peut vérifier la signature. Par exemple, un serveur, après avoir reçu cette signature, peut prouver que cette dernière est correcte mais il n'a pas la possibilité de prouver à un autre serveur sa validité. De plus, en utilisant des attributs, le serveur ne sait pas quelle personne en particuliers peut vérifier la signature *i.e.* le serveur ne connaît pas son identité. Cette application peut, entre autres, s'étendre aux données médicales pour lesquelles les personnes souhaitant vérifier une information ne peuvent le faire que si elles vérifient certaines conditions (attributs).

Comment construire une telle signature ?

Une signature utilisant les attributs et permettant de désigner, à l'aide de ces derniers, une personne pouvant vérifier cette signature fut proposée par [FWCS12]. Nous avons cherché à proposer un protocole de signature différent en essayant d'améliorer les temps de calculs. Pour cela, nous avons eu l'idée d'utiliser un protocole de chiffrement basé sur l'identité (section 2.2.1). Ce dernier nous permet de masquer la signature. La partie la plus délicate consistait à trouver comment permettre à une personne de vérifier la signature si ses attributs étaient en adéquation avec la politique requise. Par politique, nous entendons que le vérifieur doit posséder les attributs requis afin de vérifier la signature. Si ses attributs ne vérifient pas les conditions établies par la politique, l'utilisateur ne pourra pas vérifier la signature. Pour cela, nous nous sommes inspirés du schéma DIBE, présenté section 3.1.4, afin d'obtenir non pas un protocole qui *downgrade* la clé du vérifieur mais qui *upgrade* cette clé.

Pour résumé, dans notre construction, nous avons utilisé les briques cryptographiques suivantes :

— L'IBE proposé par Blazy *et al.* dans [BKP14].
— La signature de Waters [Wat05] nous permet de signer notre message pour une identité précise. Dans notre cas, les attributs seront traduits de manière unique en une identité.
— Le schéma DIBE proposé par [BGP19] pour lequel nous avons modifié leur algorithme USKDown afin d'*upgrader* la clé. Nous avons également utilisé l'idée présente dans leur article qui traduit une suite d'attributs en une identité.

Afin de traduire une liste d'attributs en une seule identité, nous avons utilisé la transformation présentée dans la section 2.2.2 que nous rappelons ici. Nous notons par $\mathbb{U}$ l'univers des attributs.

Pour tout sous-ensemble $S \subseteq \mathbb{U}$, on définit $\mathsf{id}_S \in \{0,1\}^\ell$ où le i-ème bit est défini par :

$$\mathsf{id}_S[i] := \begin{cases} 1 & \text{si } i \in S \\ 0 & \text{sinon} \end{cases}$$

Ainsi, à partir d'un ensemble d'attributs S, nous obtenons une identité qui est associée à S. Par conséquent, si l'utilisateur possède l'attribut i, il y aura un 1 à la i-ème position de l'identité, sinon se sera un 0.

Exemple 1. *Considérons l'univers des attributs suivants :*

$\mathbb{U} = \{$*"doctorant", "maître de conférence", "informatique", "astrophysique", "Limoges", "Marseille"*$\}$.

Pour un doctorant en informatique de l'université de Limoges, son identité sera de la forme : $\mathsf{id}_S = [1,0,1,0,1,0]$.

Dans notre construction, à la différence du DIBE original, on ne parle pas de *downgrade* la clé mais de l'*upgrade*. Lorsque la clé du vérifieur est générée, tous ces attributs sont utilisés. Pour vérifier la signature, il est possible que le vérifieur possède trop d'attributs par rapport à ceux requis. Ainsi, il ne pourra pas vérifier la signature même s'il possède tous les attributs nécessaires. C'est dans ce cas que l'algorithme qui permet d'*upgrade* la clé est utile. Il lui suffit de retirer les attributs en trop présents dans l'identité du vérifieur afin qu'il puisse vérifier la signature.

Exemple 2. *En reprenant l'exemple précédent, si tout doctorant inscrit en informatique dans une université française peut vérifier une signature, il devra* upgrader *son identité de la manière suivante :* $\mathsf{id}_S = [1, 0, 1, 0, 0, 0]$.

Il est important de noter que les différents schémas utilisés dans notre construction reposent sur le même IBKEM *i.e.* celui de Blazy-Kiltz-Pan. Concernant la sécurité, ce schéma est PR-ID-CPA sous l'hypothèse $k - \mathsf{MDDH}$. Cette propriété nous permettra de démontrer les différentes propriétés de sécurité de notre signature.

Transformation de Naor

La transformation de Naor permet d'obtenir un protocole de signature à partir d'un chiffrement basé sur l'identité. Cet transformation fut formalisée par Cui *et al.* dans [CFH+07].

On rappelle qu'un IBE est composé de quatre algorithmes $\mathsf{Setup}_{\mathsf{IBE}}$, KeyGen, Encrypt et Decrypt.

$\underline{\mathsf{Setup}(1^\kappa)}$:	$\underline{\mathsf{Sign}(\mathsf{sk}, \mathsf{vk}, m)}$:	$\underline{\mathsf{Verify}(\mathsf{vk}, \sigma, m)}$:
$(\mathsf{param}, \mathsf{msk}) \leftarrow \mathsf{Setup}_{\mathsf{IBE}}$	$\mathsf{id} = m$	$\mathsf{id} = m$
$\mathsf{vk} = \mathsf{param}$	$\mathsf{usk}[\mathsf{id}] \leftarrow \mathsf{KeyGen}(\mathsf{vk}, \mathsf{sk}, \mathsf{id})$	$\mathsf{usk}'[\mathsf{id}] \leftarrow \sigma$
$\mathsf{sk} = \mathsf{msk}$	$\sigma \leftarrow \mathsf{usk}[\mathsf{id}]$	$m_1 \xleftarrow{\$} \mathcal{M}$
Return $(\mathsf{vk}, \mathsf{sk})$	Return (σ, m)	$c \leftarrow \mathsf{Encrypt}(\mathsf{vk}, \mathsf{id}, m_1)$
		$m' \leftarrow \mathsf{Decrypt}(\mathsf{usk}'[\mathsf{id}], c, \mathsf{vk})$
		Return 1 si $m' = m_1$ et 0 sinon

FIGURE 4.1 – Formalisation de la transformation de Naor.

La sécurité d'une signature obtenue à l'aide de cette transformation *i.e.* la résistance aux contrefaçons est assurée par la sécurité sémantique du protocole IBE utilisé.

Nous utilisons le principe de cette transformation afin d'obtenir notre signature ABDVS.

HIBKEM

Dans la construction de la signature ABDVS, nous utilisons un HIBKEM afin de masquer la signature. L'HIBKEM utilisé est présenté dans la figure 4.2. Cette HIBKEM est basé sur l'IBE de Blazy *et al.* [BKP14], construction rappelée dans le chapitre des définitions (3.1.2). Concernant la sécurité, ce protocole est PR-ID-CPA sous l'hypothèse $k - \mathsf{MDDH}$. La propriété PR-ID-CPA implique la sécurité sémantique ainsi que l'anonymat. Ces deux propriétés sont utiles afin d'assurer la sécurité de notre construction. Dans notre construction, nous utilisons ces propriétés sous l'hypothèse SXDH. On peut remarquer que l'algorithme USKUp de notre signature est construit sur le modèle de l'algorithme USKDel de l'HIBKEM.

Présentation de la construction

Dans la figure 4.3 sont présentés les différents blocs que nous utilisons afin d'obtenir une ABDVS.

Les différents blocs sont utilisés de la manière suivante :

$\underline{\mathsf{Gen}(\mathfrak{K})}$:
$\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$
$\mathsf{sk}_{\mathsf{MAC}} = (\mathbf{B}, \mathbf{x}_0, \ldots, \mathbf{x}_\ell, x') \xleftarrow{\$} \mathsf{Gen}_{\mathsf{MAC}}(\mathfrak{K})$
$\forall i \in [\![0, \ell]\!] : \mathbf{Y}_i \xleftarrow{\$} \mathbb{Z}_p^{k\times n}$, $\mathbf{Z}_i = (\mathbf{Y}_i^\top | \mathbf{x}_i) \cdot \mathbf{A} \in \mathbb{Z}_p^{n\times k}$
$\mathbf{d}_i = \mathbf{x}_i^\top \cdot \mathbf{B}$; $E_i = \mathbf{Y}_i \cdot \mathbf{B} \in \mathbb{Z}_p^{k\times n}$
$\mathbf{y'} \xleftarrow{\$} \mathbb{Z}_p^k$; $\mathbf{z'} = (\mathbf{y'}^\top | x') \cdot \mathbf{A} \in \mathbb{Z}_p^{1\times k}$
$\mathsf{mpk} = (\mathcal{G}, [\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{[\![0,\ell]\!]}, [\mathbf{z'}]_1)$
$\mathsf{dk} := ([\mathbf{B}]_2, [\mathbf{d}_i]_2, [E_i])$
$\mathsf{msk} = (\mathsf{sk}_{\mathsf{MAC}}, (\mathbf{Y}_i)_{[\![0,\ell]\!]}, \mathbf{y'})$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{mpk}, \mathsf{id})}$:
$([\mathbf{t}]_2, [u]_2) \xleftarrow{\$} \mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, \mathsf{id})$
$\mathbf{v} = \sum_{i=0}^{\ell} f_i(\mathsf{id}) \mathbf{Y}_i \mathbf{t} + \mathbf{y'} \in \mathbb{Z}_p^k$
Pour $i = 1, ..., \ell$:
$d_i = \mathbf{x}_i^\top \mathbf{t} \in \mathbb{Z}_p$
$e_i = \mathbf{Y}_i \mathbf{t} \in \mathbb{Z}_p^k$
$\mathsf{usk}[\mathsf{id}] = ([\mathbf{t}]_2, [u]_2, [\mathbf{v}]_2) \in \mathbb{G}_2^{n+1+k}$
$\mathsf{udk}[\mathsf{id}] = ([d_i]_2, [e_i]_2)$
Return $(\mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}])$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})}$
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$
$\mathbf{c}_0 = \mathbf{A}\mathbf{r} \in \mathbb{Z}_p^{k+1}$, $\mathbf{c}_1 = (\sum_{i=0}^{l} f_i(\mathsf{id}) \mathbf{Z}_i) \cdot \mathbf{r} \in \mathbb{Z}_p^n$
$K = \mathbf{z'} \cdot \mathbf{r} \in \mathbb{Z}_p$
Return $\mathsf{K} = [K]_T$ et $C = ([\mathbf{c}_0]_1, [\mathbf{c}_1]_1) \in \mathbb{G}_1^{n+k+1}$

$\underline{\mathsf{USKDel}(\mathsf{dk}, \mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}], \mathsf{id}, \mathsf{id}_{p+1})}$:
Si $p \geq m$, alors $\perp$
$\mathsf{id}' := (\mathsf{id}_1, ..., \mathsf{id}_p, \mathsf{id}_{p+1})$
Délégation de u et $\mathbf{v}$:
$\hat{u} = u + \sum_{i=\mathsf{id}+1}^{\mathsf{id}'} f_i(\mathsf{id}') d_i$
$\hat{\mathbf{v}} = \mathbf{v} + \sum_{i=\mathsf{id}+1}^{\mathsf{id}'} f_i(\mathsf{id}') e_i$
Rerandomisation de $\hat{u}$ et de $\hat{\mathbf{v}}$:
$s' \xleftarrow{\$} \mathbb{Z}_p^n$
$\mathbf{t'} = \mathbf{t} + \mathbf{B}s' \in \mathbb{Z}_p^n$
$u' = \hat{u} + \sum_{i=0}^{\mathsf{id}'} f_i(\mathsf{id}') \mathbf{d}_i s'$
$\mathbf{v}' = \hat{v} + \sum_{i=0}^{\mathsf{id}'} f_i(\mathsf{id}') E_i s'$
Rerandomisation de d_i' et de e_i :
Pour $i = \mathsf{id}' + 1, ..., \ell$:
$d_i' = d_i + \mathbf{d}_i s' \in \mathbb{Z}_p$
$e_i' = e_i + E_i s' \in \mathbb{Z}_p^k$
$\mathsf{usk}[\mathsf{id}'] := ([\mathbf{t}']_2, [u]_2, [\mathbf{v}]_2)$
$\mathsf{udk}[\mathsf{id}'] := ([d_i']_2, [e_i']_2)$
Return $(\mathsf{usk}[\mathsf{id}'], \mathsf{udk}[\mathsf{id}'])$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, C)}$:
$K = e\left([\mathbf{c}_0]_1, \begin{bmatrix}\mathbf{v}\\ u\end{bmatrix}_2\right) / e([\mathbf{c}_1]_1, [\mathbf{t}]_2)$
Return $K \in \mathbb{G}_T$

FIGURE 4.2 – HIBKEM basé sur le tight IBE de Blazy *et al.*

$\underline{\mathsf{Setup}(1^{\mathfrak{K}})}$:
$\mathsf{mpk}, \mathsf{msk} = \mathsf{HIBKEM.Gen}(1^{\mathfrak{K}})$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}_1)}$:
$\mathsf{usk}, \mathsf{dk}, \mathsf{ek} := \mathsf{HIBKEM.USKGen}(\mathsf{msk}, \mathsf{id}_1 || 1_M)$

$\underline{\mathsf{Sign}(\mathsf{usk}, \mathsf{ek}, \mathsf{id}_1, m, \mathsf{mpk}, \mathsf{id}_2)}$:
$\sigma_1, \sigma_2 \leftarrow \mathsf{HIBKEM.USKDown}(\mathsf{usk}, \mathsf{ek}, \mathsf{id}_1 || 1_m, \mathsf{id}_1 || m)$
$C, \mathsf{K} \leftarrow \mathsf{HIBKEM.Enc}(\mathsf{mpk}, \mathsf{id}_2; \sigma_2)$
Return (C, K)

$\underline{\mathsf{Decap}(\mathsf{dk}, C)}$:
$\mathbb{F} = \bigvee_{j=1}^{k} (\bigwedge_{a \in I} a)$
Trouve $j \in [1, k]$ tel que $S_j \subseteq \mathbb{A}$
$\mathsf{uuk} \leftarrow \mathsf{USKUp}(\mathsf{usk}[\mathsf{id}], \mathsf{id}_{S_j}, \mathsf{dk})$
$K_j \leftarrow \mathsf{Dec.HIBKEM}(\mathsf{uuk}, C_j)$

FIGURE 4.3 – Vue simplifiée d'un ABDVS basé sur un HIBKEM.

— Grâce à l'HIBKEM, nous générons une clé $\mathsf{id}_1 || m$ en rétrogradant la clé de l'utilisateur id_1. En utilisant la transformation de Naor, c'est équivalent à utiliser la signature pour signer

le message m.

— IDSPHF est utilisé afin de masquer la signature.

— L'HIBKEM génère la clé du vérifieur et permet d'améliorer sa clé.

— L'algorithme USKUp permet au vérificateur désigné d'obtenir une nouvelle à partir de sa première clé et de ses attributs. Cet algorithme est analogue à l'algorithme USKDown utilisé dans le DIBE présenté section 3.1.4.

— Le premier algorithme de signature permet de générer une signature σ_1 que pourra vérifier tout utilisateur pour s'assurer que le signataire est honnête.

Concernant le modèle de sécurité, notre construction doit vérifier différentes propriétés de sécurité :

— la *correctness* comme toute signature électronique. Une signature ABDVS est correcte si pour tout $\mathfrak{K} \leftarrow \mathbb{N}$, pour toute clé $\mathsf{usk} \leftarrow \mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$, pour tout message $m \leftarrow \mathcal{M}$, on a :

$$\mathsf{Verify}(\mathsf{uuk}[\tilde{\mathsf{id}}], \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{Sign}(\mathsf{msk}, \mathsf{id}, m))) = 1.$$

— la résistance aux contrefaçons comme tout schéma de signature électronique. Une signature est non forgeable si après n requêtes de signatures valides, l'adversaire ne peut pas générer une $n+1$-ième signature valide. Dans notre protocole, nous allons prouver cette propriété en utilisant la sécurité sémantique du protocole de chiffrement utilisé (il est prouvé sémantiquement sûr sous l'hypothèse SXDH).

— la non-transférabilité qui est une propriété des *designated verifier* signatures. Un adversaire ne peut pas convaincre une tierce personne de la validité (ou de l'invalidité) d'une signature. Afin de démontrer cette propriété pour notre construction, nous allons utiliser la sécurité sémantique de l'HIBKEM sous l'hypothèse SXDH.

— la *perfect privacy* qui est une propriété des signatures à base d'attributs. Un adversaire ne peut pas déterminer avec quelle clé un message est signé. Comme pour la propriété précédente, nous utilisons la sécurité de l'HIBKEM, en particuliers l'anonymat prouvé sous l'hypothèse SXDH afin d'assurer la *perfect privacy* de notre protocole.

Remarque 4. *Si l'on compare les propriétés de sécurité que doit vérifier notre* ABDVS *avec celles présentes dans les définitions des différentes signatures, on peut constater qu'il manque la propriété de DV*-unforgeability*. Dans notre cas, cette propriété se recoupe avec celle de la résistance aux contrefaçons.*

Instanciation

Nous notons par $\mathbb{F}$ une structure d'accès associée à l'univers $\mathbb{U}$.

On retrouve bien, dans la figure 4.4, les différents blocs énoncés dans la section "Comment construire une telle signature ?". L'algorithme USKUp est l'adaptation de l'USKDown du DIBE de Blazy *et al.* L'identité notée $\tilde{\mathsf{id}}$ correspond à l'identité possédant les attributs nécessaires à la vérification de la signature. Le couplage présent dans l'algorithme Verify est une adaptation du couplage de vérification de l'IBKEM de Blazy *et al.*

Dans l'algorithme de vérification, deux équations de couplage sont utilisées :

— La première équation de couplage permet de vérifier la signature si et seulement si les attributs du vérifieur correspondent à ceux nécessaires pour vérifier la signature.

— La deuxième équation de couplage permet de vérifier qu'un utilisateur malveillant ne modifie pas l'élément aléatoire s'.

$\underline{\mathsf{Setup}(\mathfrak{K}) :}$
For $i \in \{1, 2\ell\} : z_i \xleftarrow{\$} \mathbb{Z}_p$;
$a, z' \xleftarrow{\$} \mathbb{Z}_p; y' = az'$;
$\mathsf{mpk} := (\mathcal{G}, ([z_i]_1, [z_i]_2)_{1 \le i \le 2\ell}, [a]_1, [z']_{1,2})$
$\mathsf{msk} := ((z_i)_{1 \le i \le 2\ell}), y')$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathsf{mpk}, \mathsf{id}, \perp) :}$
$s \xleftarrow{\$} \mathbb{Z}_p$
$v = y' + (\sum_1^\ell \mathsf{id}_i z_i)s$
Pour $i, \mathsf{id}[i] = 1$:
$d_i = z_i \cdot s \in \mathbb{Z}_p$;
Pour $i \in [\ell + 1, 2\ell]$:
$e_i = z_i \cdot s \in \mathbb{Z}_p$;

$\mathsf{usk}[\mathsf{id}_i] := ([s]_1, [s]_2, [v]_1)$
Return $(\mathsf{sk}[\mathsf{id}_i], \mathsf{dk} = [d_i]_1, \mathsf{ek} = [e_i]_1)$

$\underline{\mathsf{Sign}(\mathsf{mpk}, \mathsf{id}, \mathsf{sk}, \mathsf{ek}, m, \mathbb{F}) :}$
$u \xleftarrow{\$} \mathbb{Z}_p$, $s' = s + u$
$v' = v + (\sum_{i=\ell+1}^{2\ell} m_{i-\ell}\mathsf{ek}_i)s' + (\sum_{i=0}^{\ell} \mathsf{id}_i z_i)u$
$\sigma := (\sigma_{1,1}, \sigma_{1,2}, \sigma_2) = ([s']_1, [s']_2, [v']_1)$

$\mathbb{F} = \bigvee_{j=1}^{k} \mathsf{id}^{(j)}$
Pour tout $j \in [1, k]$, calcule :
$r_j \xleftarrow{\$} \mathbb{Z}_p$
$c_{j,0} = r_j$
$c_{j,1} = (\sum_{i=1}^{\ell} \mathsf{id}_i z_i) \cdot r_j$
$K_j = az' \cdot r_j + \sigma_2 \in \mathbb{Z}_p$
Return $C = [(c_{j,0}, c_{j,1})_j]_2$ $\mathsf{K} = [K_j]_T$,
$\sigma_1 = ([s']_1, [s']_2)$

$\underline{\mathsf{USKUp}(\mathsf{sk}, \tilde{\mathsf{id}}, \mathsf{dk}) :}$
Si $\neg(\mathsf{id} \preceq \tilde{\mathsf{id}})$, alors retourne $\perp$.

Posons $\mathcal{I} = \{i | \tilde{\mathsf{id}}[i] = 1 \wedge \mathsf{id}[i] = 0\}$
Upgrading la clé :
$\hat{v} = v + \sum_{i \in I} \mathsf{id}_{S_j,i} d_i \in \mathbb{Z}_p$

$\mathsf{uuk}[\tilde{\mathsf{id}}] := ([s]_1, [\hat{v}]_1)$
Return $\mathsf{uuk}[\tilde{\mathsf{id}}]$

$\underline{\mathsf{Decap}(\mathsf{mpk}, \mathsf{uuk}[\tilde{\mathsf{id}}], m, C, \mathsf{K}, \sigma_{1,1}) :}$
$\mathsf{uuk}[\tilde{\mathsf{id}}] = ([s]_1, [\hat{v}]_1)$.
$C = ([c_{j,0}]_2, [c_{j,1}]_2)$.
Vérifie quand :
$[c_0\mathsf{uuk}_2 - c_1\mathsf{uuk}_1]_T = [\mathsf{K} - (\sigma_{1,2}(\sum_{i=0}^{\ell} \mathsf{id}_i z_i + \sum_{i=\ell+1}^{2\ell} m_{i-\ell}\mathsf{ek}_i) + az')]_T$
et $[\sigma_{1,1}]_T = [\sigma_{1,2}]_T$

FIGURE 4.4 – ABDVS basé sur SXDH.

Preuves de sécurité

Commençons par montrer que notre signature est correcte.
En remplaçant chaque variable par sa valeur on obtient la simplification suivante :

$$[c_0\mathsf{uuk}_2 - c_1\mathsf{uuk}_1]_T = [\mathsf{K} - \sigma_2]_T.$$

On ne peut obtenir cette simplification si et seulement si $\mathsf{uuk}_2 = \sum_{i=1}^{l} \mathsf{id}_i z_i$ contenu dans c_1. Cette égalité est vérifiée si et seulement si les attributs du vérifieur correspondent à ceux nécessaire pour la vérification.
De même :

$$[\mathsf{K} - (\sigma_{1,2}(\sum_{i=0}^{\ell} \mathsf{id}_i z_i + \sum_{i=\ell+1}^{2\ell} m_{i-\ell}\mathsf{ek}_i) + az')]_T = [\mathsf{K} - \sigma_2].$$

On remarque qu'après avoir vérifié la signature, le vérifieur n'a pas d'information sur la signature. Il s'est seulement qu'elle est valide et qu'il possède les attributs nécessaires pour la vérifier.

La sécurité de cette construction repose sur la sécurité des protocoles utilisés. Comme nous avons pu le voir plus haut, un protocole de signature ABDVS doit être résistant aux contrefaçons, non-transférable et respecter la *perfect privacy*.

Théorème 8. *Notre construction est résistante aux contrefaçons sous l'hypothèse* SXDH.

Démonstration. L'adversaire $\mathcal{A}$ est autorisé à faire un nombre polynomial de requêtes afin d'obtenir des signatures valides. Si après ce nombre de demandes, $\mathcal{A}$ est capable de générer une signature valide alors il a prouvé que notre signature est forgeable. Dans [CFH+07], les auteurs expliquent qu'une signature obtenue à partir d'un chiffrement sur l'identité est résistante aux contrefaçons si le chiffrement utilisé est sémantique sûr. Dans notre construction, nous utilisons l'HIBKEM de Blazy *et al.* qui est sémantiquement sûr sous l'hypothèse SXDH. Ainsi, notre signature est résistante aux contrefaçons sous l'hypothèse SXDH. □

Théorème 9. *Notre signature est non-transférable sous l'hypothèse de sécurité* PR-ID-CPA *de l'*HIBKEM.

Démonstration. Soit $\mathcal{A}$ un adversaire contre la non-transferabilité de notre signature. Dans la preuve, nous allons construire un simulateur $\mathcal{B}$ qui permet à $\mathcal{A}$ d'avoir accès aux algorithmes USKGen, USKUp et Enc. Afin de démontrer que notre signature n'est pas transférable, l'adversaire va devoir "casser" la propriété PR-ID-CPA de l'HIBKEM au travers de différents jeux notés $\mathbf{G_i}$.

$\mathbf{G_0}$. Dans ce jeu nous ne modifions pas les algorithmes existants dans l'ABDVS. Il s'agit du jeu réel.

$\mathbf{G_1}$. $\mathcal{A}$ effectue m demandes aux algorithmes USKGen et Enc pour différentes identités id mais pas pour l'identité challenge id^*. Le challenger $\mathcal{B}$ répond honnêtement à ces différentes requêtes.

Pour id^*, nous remplaçons la signature σ_2 par un élément de groupe choisit aléatoirement. Nous présentons cette modification de l'algorithme Enc dans la figure 4.5.

$\underline{\mathsf{Enc}(\mathsf{id}^*) :}$ $\quad \mathbf{G_0}, \boxed{\mathbf{G_1}}$

$\mathbb{F} = \bigvee_{j=1}^{k} \mathsf{id}^{(j)}$

Pour tout $j \in [1, k]$, calcule :

$r_j \xleftarrow{\$} \mathbb{Z}_p$

$\alpha \xleftarrow{\$} \mathbb{Z}_p$

$c_{j,0} = r_j \in \mathbb{Z}_p$

$c_{j,1} = (\sum_{i=1}^{\ell} \mathsf{id}_i z_i) \cdot r_j$

$K_j = y' \cdot r_j + \sigma_2$

$\boxed{K_j = z' \cdot r_j + \alpha}$

Return $C = [(c_{j,0}, c_{j,1})_j]_1$ et $K = [K_j]_T$.

FIGURE 4.5 – Les jeux de sécurité $\mathbf{G_0}$-$\mathbf{G_1}$ pour la preuve de non-transférabilité.

En effet, cela revient à choisir la partie $c_{j,1}$ du chiffré de manière aléatoire. L'impossibilité de distinguer un chiffré aléatoire d'un vrai est assuré par la sécurité sémantique du protocole HIBKEM que nous utilisons dans notre construction. On peut donc en conclure que les jeux $\mathbf{G_1}$ et $\mathbf{G_0}$ sont indiscernables du point de vue de l'adversaire :

$$\mathsf{Adv}^{\mathbf{G_0},\mathbf{G_1}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{PR\text{-}ID\text{-}CPA}}_{\mathsf{HIBKEM}}(\mathcal{A})$$

On peut donc en conclure que notre signature est non-transférable sous l'hypothèse SXDH. □

Théorème 10. *Notre signature respecte la perfect privacy sous l'hypothèse* SXDH *grâce à la propriété de* PR-ID-CPA *de l'*HIBKEM.

Démonstration. Soit $\mathcal{A}$ un adversaire contre la *perfect privacy* de notre signature. Dans cette preuve, le challenger $\mathcal{B}$ que nous allons construire va donner à l'adversaire un accès aux algorithmes USKGen, USKUp et Enc. Afin de démontrer que notre signature respecte la *perfect privacy*, l'adversaire va devoir "casser" la propriété PR-ID-CPA de l'HIBKEM au travers de différents jeux notés $\mathbf{G_i}$.

$\mathbf{G_0}$ De même que dans la preuve précédente, dans ce jeu, nous ne modifions pas les algorithmes existants. Il s'agit du jeu réel.

$\mathbf{G_1}$ Dans ce jeu, le challenger $\mathcal{B}$ simule des clés de signature en utilisant plusieurs combinaisons d'attributs. Il donne alors à l'adversaire, aléatoirement, deux signatures générées chacune à l'aide d'une clé différente mais valide et associée aux mêmes attributs. $\mathcal{A}$ doit alors préciser s'il y a une différence entre les deux signatures *i.e.* déterminer si les signatures ont été générées à l'aide de la même clé ou pas. Ce jeu est présenté dans la figure 4.6.

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{id}) :}$	$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}_1, \mathsf{id}_2, \sigma_2, \mathbb{F}) :}$	$\mathbb{F} = \bigvee_{j=1}^{k} \mathsf{id}^{(j)}$
$t \xleftarrow{\$} \mathbb{Z}_p$	$\mathbb{F} = \bigvee_{j=1}^{k} \mathsf{id}^{(j)}$	Pour tout $j \in [1,k]$, calcule :
$v = y' + \sum_{i=1}^{\ell} \mathsf{id}_i y_i t$	Pour tout $j \in [1,k]$, calcule :	$r_j \xleftarrow{\$} \mathbb{Z}_p$
	$r_j \xleftarrow{\$} \mathbb{Z}_p$	$c_{j,0} = a \cdot r_j \in \mathbb{Z}_p$
	$c_{j,0} = a \cdot r_j \in \mathbb{Z}_p$	$c_{j,1} = (\sum_{i=1}^{\ell} \mathsf{id}_{2,i} z_i) \cdot r_j$
	$c_{j,1} = (\sum_{i=1}^{\ell} \mathsf{id}_{1,i} z_i) \cdot r_j$	$K_j = y' \cdot r_j + \sigma_2 \in \mathbb{Z}_p$
	$K_j = y' \cdot r_j + \sigma_2 \in \mathbb{Z}_p$	Return $C = [(c_{j,0}, c_{j,1})_j]_2$
	Return $C = [(c_{j,0}, c_{j,1})_j]_2$	$\mathsf{K} = [K_j]_T$
	$\mathsf{K} = [K_j]_T$	

FIGURE 4.6 – Jeux de sécurité $\mathbf{G}_0$-$\mathbf{G}_1$ pour la *perfect privacy*.

Comme rappelé dans la section 2.3.3, un adversaire ne doit pas pouvoir distinguer avec quelle clé secrète un message est signé. Dans notre protocole, cela revient à trouver pour quelle identité un chiffré a été généré. Pour réussir cette distinction, $\mathcal{A}$ doit briser l'anonymat du protocole de chiffrement utilisé. Or, l'HIBKEM que nous utilisons est anonyme sous l'hypothèse SXDH. Ainsi, nous avons :

$$\mathsf{Adv}^{\mathbf{G_0},\mathbf{G_1}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{PR-ID-CPA}}_{\mathsf{HIBKEM}}(\mathcal{A}).$$

Cette inégalité prouve la *perfect privacy* de notre protocole. □

Implementation

Une implémentation de ce protocole est proposée afin de comparer nos temps d'exécution avec l'autre protocole de signature connue. Les temps sont présentés dans le tableau 4.1.

Notre ABDVS est plus long lors de la génération des paramètres. Cependant, cette génération n'est effectuée qu'une fois, il en de même pour l'algorithme de génération de clé USKGen.

D'autre part, le temps nécessaire pour effectuer la signature et la vérification pour une identité donnée par notre protocole est largement inférieure aux temps de [FWCS12].

Par conséquent, en termes de temps globaux nous sommes plus performants comme indiqué dans la dernière ligne du tableau. On peut donc imaginer une utilisation par des serveurs *cloud* comme présenté en introduction de cette section.

Conclusion

Dans cette partie de la thèse, nous avons présenté une signature qui permet de :

Algorithme	[FWCS12]	ABDVS
Setup	22 ms	62 ms
USKGen	71 ms	112 ms
USKUp	$\emptyset$	< 1 ms
Sign	63 ms	6 ms
Verify	207 ms	16 ms
Total	363 ms	196 ms

TABLE 4.1 – Comparaison des temps des différents algorithmes de notre ABDVS avec celle de [FWCS12]

— Cibler un groupe de personnes pour vérifier une signature. Pour cela, nous avons utilisé des attributs traduits sous la forme d'une identité.

— Les vérifieurs sont désignés par l'utilisateur et ne peuvent convaincre une tierce personne de la validité de la signature.

Pour effectuer cette signature, nous avons utilisé un HIBKEM ainsi que la signature de Waters. Sous des hypothèses classiques, notre construction vérifie les propriétés de sécurité attendues *i.e.* : résistance aux collisions, respect de la *perfect privacy* et non-transférabilité.

Le point fort de cette signature comparée à la signature existantes de [FWCS12], est le temps restreint pour la vérification.

Maintenant, nous pouvons présenter une deuxième construction de signature qui permet de protéger à la fois la vie de privée de l'utilisateur et le signataire : une signature en blanc.

4.2 Signature en blanc

Les signatures sont une partie importante de la cryptographie, elles permettent de garantir l'authenticité du signataire et d'assurer l'intégrité du message. La primitive de signature connaît différentes variantes selon les cas d'usages comme par exemple, les signatures en blanc [Cha82], des signatures de groupes [Cv91, KTY04], les signatures qui utilisent des attributs [MPR11] ou encore les signatures qui permettent de désigner un vérifieur [JSI96, SZM04],... Une des utilisations les plus évidentes des signatures est le vote électronique (*e-voting*) ainsi que l'utilisation de monnaies virtuelles (*e-cash*).

Dans cette section, nous présentons une signature en blanc en tour optimal et de taille constante. En 2006, Fischlin proposa une méthode pour construire des signatures en blanc de manière optimal. Cette transformation est rappelée dans le chapitre des définitions (2.3.2). Toutefois, les signatures en blanc obtenues à partir de cette transformation sont plus grandes que la signature obtenue par le schéma sous-jacent. Notre signature est une signature en blanc en tour optimal tout en étant de taille constante *i.e.* sa taille ne dépend pas du message. Cette signature est une réponse à un problème ouvert. De plus, la sécurité de notre protocole repose sur une hypothèse standard : $k - \mathsf{MDDH}$.

Afin d'obtenir ce nouveau résultat, nous avons travaillé à partir d'une signature qui préserve la structure (celle présentée par Kiltz *et al.* dans [KPW15]). Comme résultat annexe à notre signature en blanc, nous avons obtenu une signature sur un message chiffré *randomizable*. Une des applications de cette signature est le vote électronique. Cette application possible est expliquée avec plus de détails dans la suite de ce manuscrit.

Dans la section suivante, nous présentons la signature SRC sous l'hypothèse SXDH qui est le cas $1 - \mathsf{MDDH}$ pour $k = 1$. Une version plus générale de la construction est présentée pour k quelconque dans la suite de cette section.

4.2.1 Signature sur un message chiffré *randomizable*

Dans cette partie, nous allons présenter une signature qui permet de signer sur un message chiffré *randomizable* en étant de taille constante *i.e.* la taille de la signature ne dépend pas de celle du message. Ce résultat fut présenté à la conférence internationale SECRYPT 2020 [BBCF20].

La notion de signature sur un chiffré *randomizable* fut introduite par Blazy *et al.* dans [BFPV11]. La définition de cette dernière est présentée dans la section 2.3.5. La nouvelle propriété de notre signature est le fait qu'elle soit de taille constante *i.e.* sa taille ne dépend pas de celle du message.

Notre construction est basée sur une signature qui préserve la structure de groupe (*structure preserving signature*), notée SPS présentée par Kiltz *et al.* dans [KPW15]. La notion de signature SPS ainsi que la construction utilisée sont présentées dans les sections 2.3.6 et 3.2.2 du chapitre des définitions. On rappelle simplement que dans une SPS le message est un élément de groupe. Nous pouvons maintenant présenter les détails de notre construction, sous l'hypothèse SXDH, dans la figure 4.7.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}) :}$
Return param

$\underline{\mathsf{KeyGen}_{\mathcal{E}}(\mathsf{param}) :}$
$\mathsf{dk} = h \xleftarrow{\$} \mathbb{Z}_p, \mathsf{ek} = [1, h]_1 \in \mathbb{G}_1^2,$
Return ek, dk

$\underline{\mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param}) :}$
$\mathbf{A}, \mathbf{B} \xleftarrow{\$} \mathcal{D}_2, \mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$
$\mathbf{K}_0, \mathbf{K}_1 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$
$\mathbf{C} = \mathbf{K}\mathbf{A} \in \mathbb{Z}_p^2$
$(\mathbf{C}_0, \mathbf{C}_1) = (\mathbf{K}_0\mathbf{A}, \mathbf{K}_1\mathbf{A}) \in \mathbb{Z}_p^2 \times \mathbb{Z}_p^2$
$(\mathbf{P}_0, \mathbf{P}_1) = (\mathbf{B}^\top\mathbf{K}_0,$
$\quad \mathbf{B}^\top\mathbf{K}_1) \in \mathbb{Z}_p^{1\times 2} \times \mathbb{Z}_p^{1\times 2}$
$\mathsf{sk} = (\mathbf{K}, [\mathbf{P}_0]_1, [\mathbf{P}_1]_1, [\mathbf{B}]_1)$
$\mathsf{vk} = ([\mathbf{C}_0]_2, [\mathbf{C}_1]_2, [\mathbf{C}_2]_2, [\mathbf{A}]_2)$
Return vk, sk

$\underline{\mathsf{Encrypt}(\mathsf{ek}, \perp, m) :}$
$r \xleftarrow{\$} \mathbb{Z}_p,$
$c = [r, rh + m]_1$
Return c

$\underline{\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; \mathbf{s}) :}$
$s \xleftarrow{\$} \mathbb{Z}_p, \tau \xleftarrow{\$} \mathbb{Z}_p$
$\sigma_1 = [(1, c^\top)\mathbf{K} + s(\mathbf{P}_0 + \tau\mathbf{P}_1)]_1 \in \mathbb{G}_1^{1\times 2}$
$\sigma_{\mathsf{ek}} = [(0, \mathsf{ek}^\top)\mathbf{K} + s(\mathbf{P}_0 + \tau\mathbf{P}_1)]_1 \in \mathbb{G}_1^{1\times 2}$
$\sigma_2 = [s\mathbf{B}^\top]_1 \in \mathbb{G}_1^{1\times 2}$
$\sigma_\tau = \tau \in \mathbb{Z}_p$
Return $\sigma = (\sigma_1, \sigma_{\mathsf{ek}}, \sigma_2, \sigma_\tau)$

$\underline{\mathsf{Decrypt}(\mathsf{dk}, c) :}$
Déchiffre le chiffré c en utilisant dk.

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, c, \sigma) :}$
Vérifie quand :
$[\sigma_1\mathbf{A}^\top]_T = [(1, c^\top)\mathbf{C}^\top + \sigma_2(\mathbf{C}_0^\top + \sigma_\tau\mathbf{C}_1^\top)]_T$
$[\sigma_{\mathsf{ek}}\mathbf{A}^\top]_T = [(0, \mathsf{ek}^\top)\mathbf{C}^\top + \sigma_2(\mathbf{C}_0^\top + \sigma_\tau\mathbf{C}_1^\top)]_T$

$\underline{\mathsf{WRandom}(\mathsf{vk}, \mathsf{ek}, c, \sigma) :}$
$r' \xleftarrow{\$} \mathbb{Z}_p$
Calcule $c' = c + [r', r'h]_1$ et met à jour :
$\sigma_1' = \sigma_1 + r'^\top\sigma_{\mathsf{ek}}$
$\sigma_2' = (1 + r')\sigma_2$
Return c' et $\sigma' = (\sigma_1', \sigma_2', \sigma_\tau)$

$\underline{\mathsf{WVerify}(\mathsf{vk}, \mathsf{ek}, c', \sigma')}$
$[\sigma_1'\mathbf{A}^\top]_T \stackrel{?}{=} [(1, c'^\top)\mathbf{C}^\top + \sigma_2'(\mathbf{C}_0^\top + \sigma_\tau\mathbf{C}_1^\top)]_T$

FIGURE 4.7 – SRC de taille constante reposant sur SXDH.

Comme on peut le voir dans la figures précédente 4.7, nous avons remplacé l'algorithme Random d'une SRC par WRandom. Lorsque l'on souhaite randomiser un chiffrement on désire obtenir une chiffré d'un même message mais qui n'est pas lié au chiffré précédent. Ainsi, la forme du chiffré randomisé obtenu n'est pas important. C'est dans ce sens que nous présentons notre algorithme :

— $\mathsf{WRandom}(\mathsf{vk}, \mathsf{ek}, c, \sigma, r')$: génère un chiffré c' qui chiffre le même message que c sous la clé ek ainsi qu'une signature σ' sur c' valide sous vk.

L'algorithme WVerify est l'algorithme de vérification associé à WRandom. Il permet de vérifier si la signature σ' est valide.

En ce qui concerne la sécurité, notre construction est résistante aux contrefaçons. Cette propriété est assurée par la sécurité de la construction SPS utilisée. En effet, comme nous allons le prouver, l'ajout des deux algorithmes WRandom et WVerify n'influent pas sur la résistante aux contrefaçons de la construction utilisée. Un adversaire ne pourra pas forger de signature de notre construction à partir de m signatures valides sans forger une signature SPS.

Avant de commencer les preuves de sécurité, montrons que notre signature est correcte, en commençant par montrer la *correctness* de la signature σ.

$$[\sigma_1 \mathbf{A}^\top]_T = [((1, c^\top)\mathbf{K} + s(\mathbf{P}_0 + \tau \mathbf{P}_1))\mathbf{A}^\top]_T.$$

En utilisant les définitions de $\mathbf{C}$, $\mathbf{C}_0$ et de $\mathbf{C}_1$, on retrouve le résultat attendu :

$$[\sigma_1 \mathbf{A}^\top]_T = [(1, c^\top)\mathbf{C}^\top + \sigma_2(\mathbf{C}_0^\top + \sigma_\tau \mathbf{C}_1^\top)]_T.$$

$$[\sigma_{\mathsf{ek}} \mathbf{A}^\top]_T = [((0, \mathsf{ek}^\top)\mathbf{K} + s(\mathbf{P}_0 + \tau \mathbf{P}_1))\mathbf{A}^\top]_T$$

De même que dans le cas précédent, on remplace $\mathbf{C}$, $\mathbf{C}_0$ et $\mathbf{C}_1$ par leur définition pour obtenir :

$$[\sigma_{\mathsf{ek}} \mathbf{A}^\top]_T = [(0, \mathsf{ek}^\top)\mathbf{C}^\top + \sigma_2(\mathbf{C}_0^\top + \sigma_\tau \mathbf{C}_1^\top)]_T$$

Vérifions maintenant la *correctness* de la signature randomisée.

$$\begin{aligned}
[\sigma_1' \mathbf{A}^\top]_T &= [((1, c^\top)\mathbf{K} + s(\mathbf{P}_0 + \tau \mathbf{P}_1) + r'^\top(0, \mathsf{ek}^\top)\mathbf{K} + r'^\top s(\mathbf{P}_0 + \tau \mathbf{P}_1))\mathbf{A}^\top]_T \\
&= [(1, c^\top)\mathbf{C}^\top + r'^\top(0, \mathsf{ek}^\top)\mathbf{C}^\top + (s\mathbf{B}^\top(\mathbf{K}_0 + \tau \mathbf{K}_1) + r'^\top s\mathbf{B}^\top(\mathbf{K}_0 + \tau \mathbf{K}_1))\mathbf{A}]_T \\
&= [(1, c^\top)\mathbf{C}^\top + r'^\top(0, \mathsf{ek}^\top)\mathbf{C}^\top + (\sigma_2'(\mathbf{C}_0^\top + \tau \mathbf{C}_1^\top))]_T \\
&= [(1, c'^\top)\mathbf{C}^\top + \sigma_2'(\mathbf{C}_0^\top + \sigma_\tau \mathbf{C}_1^\top)]_T.
\end{aligned}$$

Ainsi, notre signature SRC est bien correcte. Nous pouvons maintenant montrer que cette signature est résistante aux contrefaçons sous l'hypothèse SXDH.

Théorème 11. *Sous l'hypothèse* SXDH*, notre construction est résistante aux contrefaçons.*

Démonstration. Dans cette preuve, nous allons simuler étape par étape, à l'aide de jeux, la résistante aux contrefaçons de notre construction. Afin de réussir à démontrer cette propriété, nous allons utiliser le lemme 1. De plus, cette preuve est une adaptation de celle de la signature SPS de Kiltz *et al.*.

Dans la suite, les différents jeux sont notés $\mathbf{G_i}$ et les avantages associés Adv_i.

$\mathbf{G_0}$. On ne modifie aucun algorithme par rapport au protocole originale, il s'agit du jeu réel.

$\mathbf{G_1}$. Dans ce jeu, nous modifions seulement l'algorithme de vérification Verify par rapport au jeu 0. Nous remplaçons $\mathbf{C}_0$ et $\mathbf{C}_1$ par leur définition. L'équation de couplage présente dans Verify devient : $[\sigma_1 \cdot 1]_T = [(1, \mathbf{m}^\top \mathbf{K}) \cdot 1 + \sigma_2 \cdot (\mathbf{K}_0 + \tau \mathbf{K}_1)]_T$.

Supposons que $[\sigma_2 \cdot \sigma_\tau]_T = [\sigma_{\mathsf{ek}}]_T$. On peut alors remarquer que :

$$\begin{aligned}
&[\sigma_1 \cdot \mathbf{A}]_T = [(1, \mathbf{m}^\top) \cdot \mathbf{C} + \sigma_2 \cdot \mathbf{C_0} + \sigma_{\mathsf{ek}} \cdot \mathbf{C_1}]_T \\
\iff &[\sigma_1 \cdot \mathbf{A}]_T = [(1, \mathbf{m}^\top) \cdot \mathbf{KA} + \sigma_2 \cdot \mathbf{K}_0\mathbf{A} + \sigma_{\mathsf{ek}} \cdot \mathbf{K}_1\mathbf{A}]_T \\
\iff &[\sigma_1 \cdot 1]_T = [(1, \mathbf{m}^\top) \cdot \mathbf{K} + \sigma_2 \cdot \mathbf{K}_0 + \sigma_{\mathsf{ek}} \cdot \mathbf{K}_1]_T \\
\iff &[\sigma_1 \cdot 1]_T = [(1, \mathbf{m}^\top) \cdot \mathbf{K} + \sigma_2 \cdot (\mathbf{K}_0 + \tau\mathbf{K}_1)]_T
\end{aligned}$$

Pour tout couple (m, σ) qui vérifie Verify mais pas Verify^*, l'expression $\sigma_1 - ((1, m^\top)\mathbf{K} + \sigma_2\mathbf{K}_0 + \sigma_3\mathbf{K}_1) \in \mathbb{G}_1^{1\times(k+1)}$ est un vecteur non nul du noyau de la matrice $\mathbf{A}$, qui est difficile à calculer sous l'hypothèse $\mathcal{D}_k$-KerMDDH dans $\mathbb{G}_2$. Cela signifie que l'on a :

$$|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \mathsf{Adv}^{kmddh}_{\mathcal{D}_k,\mathsf{Setup}}(\mathcal{B}_0).$$

$\mathbf{G_2}$. Soit $\tau_1, ..., \tau_\mathcal{Q}$ les tags aléatoires utilisés pour les $\mathcal{Q}$ premières requêtes effectuées par l'adversaire $\mathcal{A}$ à l'oracle OSign. Nous arrêtons si les $\tau_1, ..., \tau_\mathcal{Q}$ ne sont pas tous distincts. Ainsi, on obtient :

$$\mathsf{Adv}_2 \geq \mathsf{Adv}_1 - \mathcal{Q}^2/2p.$$

$\mathbf{G_3}$. On pose $\tau_{\mathcal{Q}+1}{:=}\tau^*$. Maintenant on choisit $i^* \xleftarrow{\$} [\mathcal{Q}+1]$ et on arrête le jeu si i^* n'est pas le plus petit indice de i pour lequel $\tau^* = \tau_i$. Dans la suite de cette preuve, on se concentre sur le cas où on n'arrête pas la preuve *i.e* $\tau^* = \tau_{i^*}$ et $\tau_1, ..., \tau_{i^*-1}$ sont tous différents de τ^*. Pour un τ donné, OSign peut vérifier quand τ^* est égal à τ. Pour les autres $i^* - 1$ requêtes, on répond "NON", en commençant à la i^*-ième requête, on connaît τ^*.

Ainsi, nous obtenons :

$$\mathsf{Adv}_3 \geq \frac{1}{Q+1}\mathsf{Adv}_2.$$

Le quatrième jeu est découpé en deux sous-jeux. Dans le premier nous allons simuler la signature σ_1. Dans le second, c'est σ_{ek} qui est modifiée.

$\mathbf{G_{4.1}}$. Dans ce jeu, nous allons modifier OSign en OSign_1^* :

$\underline{\mathsf{OSign}_1^* :}$
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q; \tau \xleftarrow{\$} \mathbb{Z}_p; \mu_1 \xleftarrow{\$} \mathbb{Z}_p$
$\sigma_1{:=}[(1, C_1, C_2)\mathbf{K} + \mu_1\mathbf{A}^\perp + \mathbf{r}^\top(\mathbf{P_0} + \tau\mathbf{P_1})]_1 \in \mathbb{G}_1^{1\times 2}$
$\sigma_{\mathsf{ek}}{:=}[(0, 1, \mathsf{ek})\mathbf{K} + \mathbf{r}^\top(\mathbf{P_0} + \tau\mathbf{P_1})]_1 \in \mathbb{G}_1^{1\times 2}$
$\sigma_2{:=}[\mathbf{r}^\intercal\mathbf{B}^\intercal]_1 \in \mathbb{G}_1^{1\times 2}$,
$\sigma_\tau{:=}\tau \in \mathbb{Z}_p$
$\mathtt{Return}(\sigma_1, \sigma_{\mathsf{ek}}, \sigma_2, \sigma_\tau)$

Ici, $\mathbf{A}^\perp \in^{1\times(k+1)}$ est un vecteur non-nul du noyau de la matrice $\mathbf{A}$ tel que : $\mathbf{A}^\perp\mathbf{A} = 0$. Nous allons appliquer le lemme 1 afin de montrer que :

$$|\mathsf{Adv}_3 - \mathsf{Adv}_{4.1}| \geq 2\mathcal{Q}\mathsf{Adv}^{mddh}_{\mathcal{D}_k,\mathsf{Setup}}(\mathcal{B}_1) + \mathcal{Q}/p.$$

Nous choisissons $\mathbf{K}$ et utilisons $\mathcal{O}_b$ pour simuler soit OSign soit OSign_1^* et $\mathcal{O}^*$ pour simuler Verify comme suit :

— Pour la i-ième demande de signature où $i \neq i^*$, on demande à $\mathcal{O}_b$ à $\tau \xleftarrow{\$} \mathbb{Z}_q$ afin d'obtenir :

$$(\sigma_1', \sigma_2){:=}([b\mu_1\mathbf{A}^\perp + \mathbf{r}^\top(\mathbf{P}_0 + \tau\mathbf{P}_1)]_1, [\mathbf{r}^\top\mathbf{B}^\top]_2)$$

avec $b \xleftarrow{\$} \{0, 1\}$ et on renvoie :

$$\sigma_1{:=}([(1, \mathbf{m}\top)\mathbf{K}]_1 \cdot \sigma_1', \sigma_{\mathsf{ek}}, \sigma_2, \sigma_3, \sigma_4).$$

— Pour la i^*-ième requête, celle du challenge, où $i^* \leq \mathcal{Q}$, on utilise honnêtement l'algorithme de signature Sign.

— Pour Verify^*, on demande $\mathcal{O}^*$ sur τ^*pour obtenir $[\mathbf{K}_0 + \tau^*\mathbf{K}_1]_2$. Ce dernier est suffisant pour simuler les requêtes effectuées à Verify^* en calculant $[\sigma \cdot (\mathbf{K}_0 + \tau^*\mathbf{K}_1)]_T$.

$\mathbf{G_{4.2}}$.

Ce jeu est construit sur le même modèle que le précédent. Ici, nous devons modifier σ_{ek} comme suit :

$\underline{\mathsf{OSign}_2^*}$:
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q; \tau \xleftarrow{\$} \mathbb{Z}_q; \mu_1, \mu_2 \xleftarrow{\$} \mathbb{Z}_q$
σ_1:=$[(1, C_1, C_2)\mathbf{K} + \mu_1\mathbf{A}^\perp + \mathbf{r}^\top(\mathbf{P_0} + \tau\mathbf{P_1})]_1 \in \mathbb{G}_1^{1\times 2}$
σ_{ek}:=$[(0, 1, \mathsf{ek})\mathbf{K} + \mu_2\mathbf{A}^\perp + \mathbf{r}^\top(\mathbf{P_0} + \tau\mathbf{P_1})]_1 \in \mathbb{G}_1^{1\times 2}$
σ_2:=$[\mathbf{r}^\intercal\mathbf{B}^\intercal]_1 \in \mathbb{G}_1^{1\times 2}$,
σ_τ:=$\tau \in \mathbb{Z}_p$
$\mathtt{Return}(\sigma_1, \sigma_{\mathsf{ek}}, \sigma_2, \sigma_\tau)$

Afin de pouvoir appliquer le lemme 1, on simule les oracles de cette manière :

— Pour $i \neq i^*$, nous avons :

$$(\sigma_1', \sigma_{\mathsf{ek}}', \sigma_2) := ([b\mu_1\mathbf{A}^\perp + \mathbf{r}^\top(\mathbf{P}_0 + \tau\mathbf{P}_1)]_1, [b\mu_2\mathbf{A}^\perp + \mathbf{r}^\top(\mathbf{P}_0 + \tau\mathbf{P}_1)]_1, [\mathbf{r}^\top\mathbf{B}^\top]_2).$$

On renvoie : $(\sigma_1$:=$[(1, \mathbf{m}\top)\mathbf{K}]_1 \cdot \sigma_1', \sigma_{\mathsf{ek}}$:=$[(1, \mathbf{m}\top)\mathbf{K}]_1 \cdot \sigma_{\mathsf{ek}}', \sigma_2, \sigma_4)$.

— Pour le message challenge, nous utilisons honnêtement Sign.

— Verify^* est simuler comme dans le $\mathbf{G_{4.1}}$.

Ainsi, nous avons réussi à construire dans les jeux $\mathbf{G_{4.1}}$ et $\mathbf{G_{4.2}}$ deux "distinguisher" différents pour le lemme 1

$\mathbf{G_5}$. Dans ce jeu, nous modifions la matrice $\mathbf{K}$ dans l'algorithme USKGen. On change $\mathbf{K}$ en $\mathbf{K} = \mathbf{K'} + \mathbf{u}\mathbf{A}^\perp$ avec $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times(k+1)}$ et $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{n+1}$. Par conséquent, $\mathbf{u}\mathbf{A}^\perp$ est masqué par la matrice $\mathbf{K'}$ et $\mathbf{K}$ est toujours uniformément aléatoire, la matrice $\mathbf{K'}$ étant aléatoire. Ainsi, les jeux 5 et 4.2 sont identiques et on obtient alors :

$$\mathsf{Adv}_5 = \mathsf{Adv}_{4.2}$$

.

Maintenant, que nous avons effectués ces différents jeux, nous devons majorer Adv_5.

— On a $C = \mathbf{K}\mathbf{A} = (\mathbf{K'} + \mathbf{u}\mathbf{A}^\perp)\mathbf{A} = \mathbf{K'}\mathbf{A}$. Le vecteur $\mathbf{u}$ est donc totalement masqué.

— OSign_2^* prend en entrée $(C_1, C_2, \mathsf{ek}, \tau)$ avec $\tau \neq \tau^*$ et renvoie : $(1, C_1, C_2)(\mathbf{K'}+\mathbf{u}\mathbf{A}^\perp)+\mu_1\mathbf{A}^\perp$ et $(0, 1, \mathsf{ek}) + (\mathbf{K'} + \mathbf{u}\mathbf{A}^\perp) + \mu_2\mathbf{A}^\perp$. Le vecteur $\mathbf{u}$ est masqué tant que les sorties sont distribuées comme suit : $(1, C_1, C_2)\mathbf{K'} + \mu_1\mathbf{A}^\perp$ et $(0, 1, \mathsf{ek})\mathbf{K} + \mu_2\mathbf{A}^\perp$.

— Si OSign_2^* reçoit τ^*, il laisse filtrer $(1, C_1, C_2)\mathbf{K'} + \mu_1\mathbf{A}^\perp$ et $(1, \mathsf{ek})\mathbf{K} + \mu_2\mathbf{A}^\perp$, qui sont masqués par $(1, C_1, C_2)\mathbf{u}$ et par $(0, 1, \mathsf{ek})\mathbf{u}$.

L'adversaire doit correctement calculer $(1, C_1^*, C_2^*)(\mathbf{K'} + \mathbf{u}\mathbf{A}^\perp)$ et $(0, 1, \mathsf{ek})(\mathbf{K'} + \mathbf{u}\mathbf{A}^\perp)$ pour que l'algorithme Verify^* valide la signature σ^* qu'il forge sur $(1, C_1^*, C_2^*)$ et sur $(0, 1, \mathsf{ek}^*)$. Comme $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, $(1, C_1^*, C_2^*)\mathbf{u}$ et $(0, 1, \mathsf{ek}^*)\mathbf{u}$ le sont dans $\mathbb{Z}_p$.

Avec $(1, C_1^*, C_2*)\mathbf{u}$ et $(0, 1, \mathsf{ek}^*)$ donnés pour tout $(C_1^*, C_2^*) \neq (C_1, C_2)$ et $\mathsf{ek}^* \neq \mathsf{ek}$ choisit en fonction des données, $(1, C_1^*, C_2^*)\mathbf{u}$ et $(0, 1, \mathsf{ek}^*)\mathbf{u}$ sont uniformément aléatoire sur $\mathbb{Z}_p$ du point de vue de l'adversaire. Ce dernier ne peut donc pas distinguer les deux distributions. On a donc : $\mathsf{Adv}_5 \leq 1/p$. En utilisant les majorations successives des avantages de l'adversaire dans les différents jeux, on peut conclure que $\mathcal{A}$ ne peut pas forger de signature. □

SRC *extractable*

La notion de signature *extractable* fut présentée dans le même article que les signatures SRC [BFPV11].

À partir d'une signature SRC il est possible d'obtenir une nouvelle signature valide σ' sur le message m. Dans ce cas, on dit que la signature est *extractable*. Pour cela, on ajoute l'algorithme suivant à ceux déjà utilisés par une SRC :

— $\mathsf{SEDecrypt}(\mathsf{dk}, \mathsf{vk}, \sigma)$: génère une nouvelle signature σ' avec une clé de déchiffrement dk, une clé de vérification vk et une signature σ sur le message m.

Une telle signature peut avoir une application dans le vote électronique comme expliqué dans le paragraphe suivant.

Supposons qu'il existe un protocole de signature $\mathcal{S}$ tel que Setup_S, Setup_E sont respectivement associé à la génération des paramètres de la signature et du chiffrement. De même, on pose KeyGen_S et KeyGen_E qui génère les clés associées à la signature et au chiffrement.

Avec $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}_S(\mathsf{param})$, $m \in \mathcal{M}$, une valeur aléatoire $r \in \mathcal{R}$, $s \in \mathcal{R}$, $c = \mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}, m; r)$ et $\sigma = \mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; s)$, la sortie σ' de $\mathsf{SEDecrypt}(\mathsf{dk}, \mathsf{vk}, \sigma)$ est une signature valide sur m sous la clé vk si $\mathsf{Verify}_{\mathcal{S}}(\mathsf{vk}, m, \sigma')$.

On dit qu'un protocole de signature SRC est *extractable* si les assertions suivantes sont vérifiées.

Un protocole de signature SRC *extractable* se déroule en plusieurs étapes comme le montre la figure 4.8. Comme le montre ce schéma, on peut remarquer que l'étape de signature et celle de chiffrement sont commutatives.

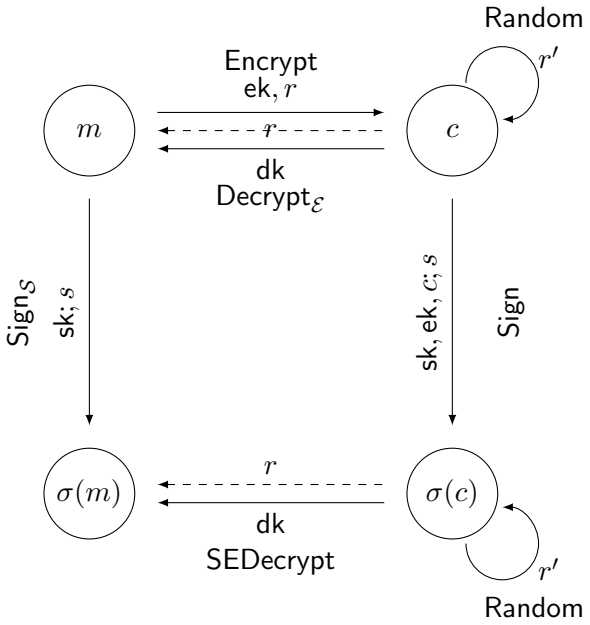


FIGURE 4.8 – SRC *extractable*.

L'utilisateur $\mathcal{U}$ chiffre un message m. Il demande ensuite une signature σ sur le chiffré c.

La personne détenant la clé de déchiffrement dk peut alors déchiffrer c à partir de (c, σ) afin de retrouver le message m. En utilisant le nouvel algorithme SEDecrypt, il peut également obtenir une nouvelle signature σ' sur c.

Une application possible : le vote électronique

Une des applications de notre schéma de signature sur un chiffré *randomizable* est le vote électronique.

En général, les protocoles utilisés dans le vote électronique reposent sur les chiffrements homomorphes [Pai99]. Ces chiffrements permettent d'effectuer des opérations sur des chiffrés et dont le résultat est toujours un chiffré. Une solution que nous proposons est l'utilisation de notre SRC optimale.

Un des risques que l'on peut rencontrer dans l'utilisation du vote électronique et la vente de vote. En effet, un électeur peut prouver qu'il a voté pour tel ou tel candidat en révélant à l'acheteur la valeur aléatoire utilisé lors du chiffrement. Pour éviter ce problème, le serveur doit être capable de randomiser le chiffré du vote et adapter la signature associée. Notre signature SRC permet de faire cela comme le montre le schéma suivant.

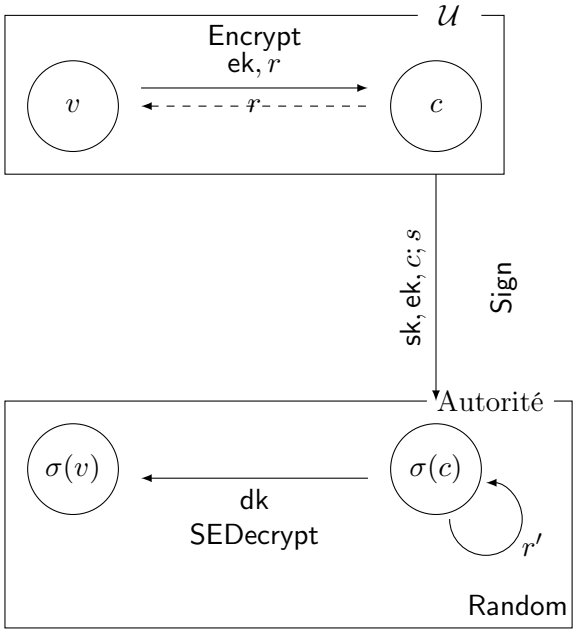


FIGURE 4.9 – SRC *extractable* utilisée pour le vote électronique.

En utilisant notre protocole de signature, l'électeur chiffre son vote v en c puis effectue la signature σ sur c. Maintenant, le centre recevant le vote et la signature peut les randomiser. L'aléatoire utilisé pour obtenir c' est inconnu de l'électeur et la nouvelle signature σ' est valide car elle ne peut pas être forger par le centre d'après les propriétés de notre signature.

Construction Générique de la signature SRC

La signature SRC qui vient d'être présentée repose sur l'hypothèse SXDH. Une généralisation de cette signature est possible sous l'hypothèse $\mathcal{D}_k - \mathsf{MDDH}$. De même que pour la construction basée sur SXDH, elle est basée sur la signature SPS de [KPW15].

Concernant la sécurité, cette signature est résistante aux contrefaçons. Pour le prouver, il est possible de réutiliser les jeux de sécurité utilisés pour la construction basée sur SXDH.

Maintenant que nous avons présenté notre protocole de signature de document chiffré en un nombre optimal d'échange, nous pouvons introduire notre signature en blanc en tour optimal.

4.2.2 Signature en blanc

La définition de signature en blanc ou *blind signature* a été présentée dans la section 2.3.6 du chapitre des définitions. Une signature en blanc est une signature qui permet de ne pas dévoiler au signataire des informations sur le message qu'il signe. Une des applications possibles, est le

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}) :}$
Return param

$\underline{\mathsf{KeyGen}(\mathsf{param}) :}$
$\mathsf{ek} = [\mathbf{H}]_1 \xleftarrow{\$} \mathbb{G}_1^{(k+1)\times k}$, $\mathsf{dk} = H \in \mathbb{Z}_p^{(k+1)\times k}$
Return ek, dk

$\underline{\mathsf{KeyGen}(\mathsf{param}) :}$
$\mathbf{A}, \mathbf{B} \xleftarrow{\$} \mathcal{D}_{k+1}$
$\mathbf{K}_0, \mathbf{K}_1 \xleftarrow{\$} \mathbb{Z}_p^{k+1\times k+1}$
$\mathbf{C} = \mathbf{K}\mathbf{A} \in \mathbb{Z}_p^{k+1}$
$(\mathbf{C}_0, \mathbf{C}_1) = (\mathbf{K}_0\mathbf{A}, \mathbf{K}_1\mathbf{A}) \in (\mathbb{Z}_p^{k+1})^2$
$(\mathbf{P}_0, \mathbf{P}_1) = (\mathbf{B}^\top\mathbf{K}_0, \mathbf{B}^\top\mathbf{K}_1) \in (\mathbb{Z}_p^{1\times(k+1)})^2$
$\mathsf{sk} = (\mathbf{K}, [\mathbf{P}_0]_1, [\mathbf{P}_1]_1, [\mathbf{B}]_1)$
$\mathsf{vk} = ([\mathbf{C}_0]_2, [\mathbf{C}_1]_2, [\mathbf{C}_2]_2, [\mathbf{A}]_2)$
Return vk, sk

$\underline{\mathsf{Encrypt}(\mathsf{ek}, \perp, m) :}$
$r \xleftarrow{\$} \mathbb{Z}_p$
Return $c = [r, rh + m]_1$

$\underline{\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; \mathbf{s}) :}$
$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k, \tau \xleftarrow{\$} \mathbb{Z}_p$
$\sigma_1 = [(1, c^\top)\mathbf{K} + \mathbf{s}^\top(\mathbf{P}_0 + \tau\mathbf{P}_1)]_1 \in \mathbb{G}_1^{1\times k+1}$
$\sigma_{\mathsf{ek}} = [(0, \mathbf{H}^\top)\mathbf{K} + \mathbf{s}^\top(\mathbf{P}_0 + \tau\mathbf{P}_1)]_1 \in \mathbb{G}_1^{k\times k+1}$
$\sigma_2 = [\mathbf{s}^\top\mathbf{B}^\top]_1 \in \mathbb{G}_1^{1\times k+1}$, $\sigma_\tau = \tau \in \mathbb{Z}_p$
Return $\sigma = (\sigma_1, \sigma_{\mathsf{ek}}, \sigma_2, \sigma_\tau)$

$\underline{\mathsf{Decrypt}(\mathsf{ek}, c) :}$
Return $[m]_1 = c_2/c_1^{\mathsf{dk}} = [rh + m - rh]_1$

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, c, \sigma) :}$
$[\sigma_1\mathbf{A}^\top]_T \stackrel{?}{=} [(1, c^\top)\mathbf{C}^\top + \sigma_2(\mathbf{C}_0^\top + \sigma_\tau\mathbf{C}_1^\top)]_T$
$[\sigma_{\mathsf{ek}}\mathbf{A}^\top]_T \stackrel{?}{=} [(0, \mathsf{ek}^\top)\mathbf{C}^\top + \sigma_2(\mathbf{C}_0^\top + \sigma_\tau\mathbf{C}_1^\top)]_T$

FIGURE 4.10 – Construction Générique de la signature SRC.

transfert d'argent : la banque ne sait pas le montant de la transaction mais peut en assurer son authenticité.

À l'aide de la signature SRC présentée dans la partie précédente, nous allons pouvoir construire une signature en blanc en un tour et de taille constante. En un tour signifie qu'il y a un seul envoi de données de la part du signataire et un seul également de la part de l'utilisateur. De taille constante signifie que la taille des données échangées ne dépend pas de la taille du message. Afin d'obtenir ces deux propriétés, nous utilisons l'algorithme WRandom, présenté dans la figure 4.7, pour randomiser une signature sur un chiffré. Cette manipulation nous permet d'obtenir une signature en blanc.

Sur l'idée présentée par Marc Fischlin dans [Fis06], nous utilisons une preuve *zero-knowledge* afin de vérifier la validité de la signature extraite. Cette transformation est brièvement présentée dans la section 2.3.2. La définition ainsi que les propriétés des preuves de connaissances ont été énoncées dans le chapitre des définitions, section 2.1.3.

Cependant, voici un point sur une notion que nous utilisons dans notre construction : les *commits* Groth-Sahai. La notion de *commits* ou de mise en gage fut présentée dans le chapitre des définitions, section 2.1.3.

Groth-Sahai *commitments*

Les *commitments* que nous présentons dans cette section sont présentés dans l'article [GS08].

Initialisation. On utilise un groupe bilinéaire de la forme : $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. La clé de *commitment* pour le groupe $\mathbb{G}_1$ est $\mathbf{u} = (\mathbf{u_1}, \mathbf{u_2})$. On l'initialise à l'aide des valeurs aléatoires $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$ de la manière suivante :

$$\mathbf{u_1} = [1, \alpha]_1 \text{ et } \mathbf{u_2} = [\beta, \alpha\beta]_1$$

.

Le *commitment* est *binding* avec la clé $\mathbf{u_2}$ et il peut devenir *hiding* si l'on définit : $\mathbf{u_2} = \beta\mathbf{u_1} - (1, g_1)$. Plus d'informations sur ces notions (*binding* et *hiding*) sont disponibles dans la section 2.1.3 du chapitre des définitions. Que l'on choisisse la première ou la deuxième forme de $\mathbf{u_2}$, $\mathbf{u}$ est sémantiquement sûre sous l'hypothèse DDH.

La clé de *commitment* $\mathbf{v}$ pour le groupe $\mathbb{G}_2$ peut être définie de manière analogue.

***Commitment* d'un élément de groupe.** Soit $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$. Pour mettre en gage un élément de groupe $\mathcal{X} \in \mathbb{G}_1$, on pose :

$$\mathcal{C}(\mathcal{X}) = [r_1 u_{1,1} + r_2 u_{2,1}, \mathcal{X} + r_1 u_{1,2} + r_2 u_{2,2}]_1.$$

***Commitment* d'un élément de $\mathbb{Z}_p$.** Soit $r \xleftarrow{\$} \mathbb{Z}_p$. Pour mettre en gage le vecteur $x \in \mathbb{Z}_p$, on pose :

$$\mathcal{C}'(x) = [r u_{1,1} + x u_{2,1}, r u_{1,2} + x(u_{2,2} + 1)]_1.$$

Selon la forme de la clé $\mathbf{u_2}$, le *commitment* est *biding* ou *hiding*.
Sous l'hypothèse SXDH, les deux instantiations des clés de *commitment* sont indistinguables.

Preuves Groth-Sahai. Une preuve Groth-Sahai est un couple d'éléments $(\pi, \theta) \in \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$. Ces éléments sont construits pour aider à vérifier les équations de couplage sur les *commitments*. Être capable de produire un couple d'éléments (π, θ) valide, signifie que l'on connaît les messages en clair qui vérifient les bonnes relations. Nous notons $\langle \mathcal{X} \rangle_1$ pour un élément de $\mathbb{G}_1$ mis en gage et $\langle x \rangle_2$ pour un scalaire x de $\mathbb{G}_2$ mis en gage.

Dans notre construction, nous allons utiliser les deux formes d'équations suivantes :

— Équations linéaires de couplage dans le groupe $\mathbb{G}_1$: $[\sum_i X_i \mathcal{B}_\mathbf{i}]_T = [t]_T$ est utilisé pour prouver que les éléments de groupe mis en gage $[X_i]_1$ vérifie des équations de couplage avec les vecteurs constants $[\mathcal{B}_\mathbf{i}]_2$ égaux à des éléments du groupe $\mathbb{G}_T$. Dans ce cas particulier, la preuve est composée d'un seul élément $\theta \in \mathbb{G}_2^2$.

— Équations de multiplication multi-scalaires : $[y\mathcal{A}]_1 = [\mathcal{X}]_1$ est utilisé pour prouver que le scalaire mis en gage dans $\mathbb{G}_2$ est le logarithme discret de $\mathcal{X}$ mis en gage dans in $\mathbb{G}_1$.

Maintenant que nous avons vu les formes de preuves Groth-Sahai que nous allons utiliser dans notre construction, nous pouvons la présenter.

Construction

Avant de présenter les détails de notre construction, la figure 4.12 présente une vue d'ensemble de cette dernière. Notamment, elle permet de mettre en évidence l'unique interaction entre l'utilisateur et le signataire.

De même que pour la signature précédente, on peut représenter notre signature en blanc sous la forme d'un schéma.

La figure 4.13 présente notre construction qui utilise les différentes briques présentées précédemment.

La notation $\mathsf{Encrypt}_{EG}$ correspond à l'utilisation du chiffrement de ElGamal, $\mathsf{Setup}_{\mathsf{SRC}}$ (respectivement $\mathsf{Sign}_{\mathsf{SRC}}$) à l'utilisation du Setup (respectivement du protocole de signature) de notre signature SRC. On rappelle que les preuves à divulgation nulle de connaissance, notées NIZK sont définies dans la section 2.1.3.

Nous pouvons constater que notre signature est bien en tour optimal. En effet, l'utilisateur envoie le message chiffré au signataire. Ce dernier le signe en utilisant notre protocole de SRC et le renvoie à l'utilisateur. Ce dernier doit alors vérifier si la signature SRC (notée σ_{SRC}) est valide . Si elle l'est, il la randomise à l'aide de WRandom. Il termine avec une preuve de connaissance afin de prouver que cette signature randomisée est bien valide.

En ce qui concerne la *correctness* de cette construction, l'algorithme qui vérifie si la signature est valide est l'algorithme de vérification de la signature précédente *i.e.* la SRC. Comme cette signature est correcte, nous pouvons en déduire que celle-ci l'est également.

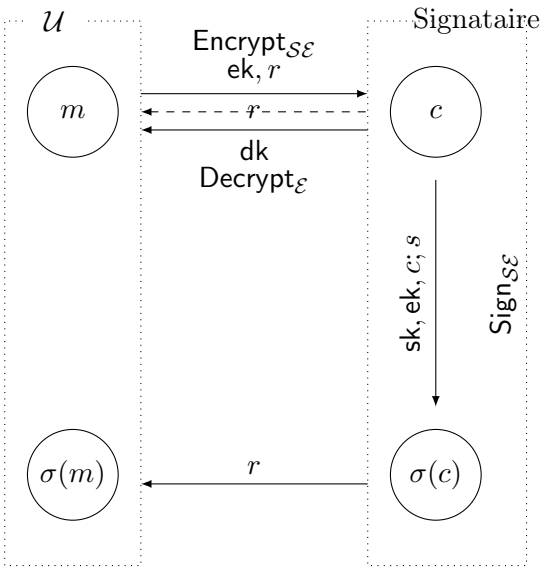


FIGURE 4.11 – Schématisation de notre signature en blanc.

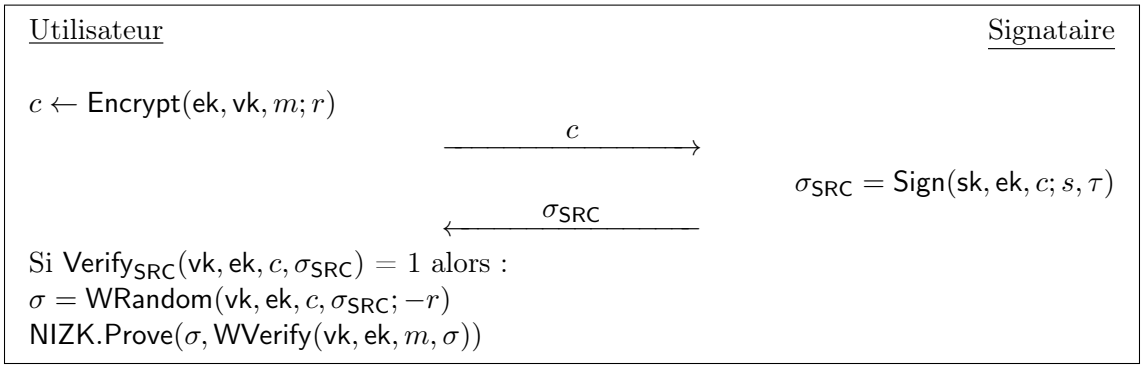


FIGURE 4.12 – Vue simplifiée de notre signature en blanc.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}) :}$
$\mathsf{param} \leftarrow \mathsf{Setup}_{\mathsf{SRC}}$
Return param

$\underline{\mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param}) :}$
$\mathsf{vk}, \mathsf{sk} \leftarrow \mathsf{KeyGen}_{S}(\mathsf{param})$
Return vk, sk

$\underline{\mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}, m; r) :}$
$c \leftarrow \mathsf{Encrypt}_{EG}$
Return c

$\underline{\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; \mathbf{s}, \tau) :}$
$\sigma_{\mathsf{SRC}} \leftarrow \mathsf{Sign}_{\mathsf{SRC}}$
Return σ_{SRC}

$\underline{\mathsf{Verify}_{\mathsf{SRC}}(\mathsf{vk}, \mathsf{ek}, c, \sigma_{\mathsf{SRC}}) :}$

Vérifie la validité de σ_{SRC}
Return 1 ou 0

$\underline{\mathsf{Random}(\mathsf{vk}, \mathsf{ek}, \sigma_{\mathsf{SRC}}) :}$
$\sigma \leftarrow \mathsf{WRandom}(\mathsf{vk}, \mathsf{ek}, c, \sigma_{\mathsf{SRC}}; r)$
Return σ

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, m, \sigma) :}$
$\mathsf{NIZK.Prove}(\sigma, \mathsf{WVerify}(\mathsf{vk}, \mathsf{ek}, m, \sigma))$
Return 1 ou 0

FIGURE 4.13 – Signature en blanc.

Nous allons prouver les deux propriétés de sécurité que doit vérifier une signature en blanc :

— la *blindness* : un signataire malicieux, qui a signé deux message m_0 et m_1, ne peut pas désigner lequel des deux messages il a signé en premier.

— la résistance aux contrefaçons : après un nombre polynomial de requêtes, l'adversaire ne doit pas être capable de créer, de forger une signature valide.

Commençons par énoncer et démontrer la propriété de *blindness* de notre construction.

Théorème 12. *Notre blind signature est blind sous l'hypothèse de sécurité* SXDH.

Démonstration. Afin d'effectuer la preuve de *blindness*, nous allons procéder par une succession de jeu de sécurité. Notons Adv_i l'avantage de l'adversaire $\mathcal{A}$ dans le jeu i. Ces différents jeux sont présentés par la figure 4.14.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}) :}$	$G_0 - G_2$	$\underline{\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; \mathbf{s}, \tau) :}$	$G_0 - G_2$
$\mathsf{param} \leftarrow \mathsf{Setup}_{\mathsf{SRC}}$		$\sigma_{\mathsf{SRC}} \leftarrow \mathsf{Sign}_{\mathsf{SRC}}$	
Return param		Return σ_{SRC}	
$\underline{\mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param}) :}$	$G_0 - G_2$	$\underline{\mathsf{Verify}_{\mathsf{SRC}}(\mathsf{vk}, \mathsf{ek}, c, \sigma_{\mathsf{SRC}})}$	$G_0 - G_2$
$\mathsf{vk}, \mathsf{sk} \leftarrow \mathsf{KeyGen}_S(\mathsf{param})$		Vérifie la validité de la signature σ_{SRC}	
Return vk, sk		Return 1 ou 0	
		$\underline{\mathsf{Random}(\mathsf{vk}, \mathsf{ek}, \sigma_{\mathsf{SRC}}) :}$	$G_0, \boxed{G_1}, G_2$
$\underline{\mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}, m; r) :}$	G_0, G_1, G_2	$\sigma \leftarrow \mathsf{WRandom}(\mathsf{vk}, \mathsf{ek}, \sigma_{\mathsf{SRC}})$	
$c \leftarrow \mathsf{Encrypt}_{EG}$		$\boxed{\sigma \xleftarrow{\$} \mathcal{S}}$	
$v \xleftarrow{\$} \mathcal{M}$		Return σ	
$c \leftarrow \mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}, v; r)$			
Return c		$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, m, \sigma) :}$	$G_0, \boxed{G_1}, G_2$
		$\mathsf{NIZK.Prove}(\sigma, \mathsf{WVerify}(\mathsf{vk}, \mathsf{ek}, \sigma))$	
		$\boxed{\mathsf{NIZK.SimulateProve}(\sigma, \mathsf{WVerify}(\mathsf{vk}, \mathsf{ek}, m, \sigma))}$	
		Return 1 ou 0	

FIGURE 4.14 – Jeux de sécurité pour la preuve de *blindness*.

$\mathbf{G_0}$. Dans ce jeu, il est important de noter que l'on ne modifie pas les algorithmes du protocole. Pendant les prochains jeux, l'adversaire, $\mathcal{A}$, va proposer deux messages : m_0 et m_1 au challenger.

$\mathbf{G_1}$. Dans ce jeu, nous modifions les preuves par une preuve simulée grâce à la propriété *Zero-Knowledge* des systèmes de preuves de Groth-Sahai. Ainsi, comme on peut le voir dans la figure 4.14, nous avons remplacé l'algorithme NIZK par un simulé. Ainsi, nous obtenons :

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \mathsf{Adv}^{\mathsf{SXDH}}_{\mathcal{GS}}.$$

$\mathbf{G_2}$. Dans la fonction Encrypt, les chiffrés de m_0 et de m_1 sont remplacés par le chiffrement d'un élément de groupe alétoire *i.e* le challenger chiffre un élément de groupe aléatoire noté v au lieu de chiffrer m_b, avec $b \in \{0, 1\}$. En utilisant la propriété de sécurité sémantique du protocole de chiffrement ElGamal (cf. section 3.1.1), nous obtenons :

$$\mathsf{Adv}_1 \leq \mathsf{Adv}_2 + \mathsf{Adv}^{\mathsf{SXDH}}_{\mathsf{EG}}.$$

Grâce à la sécurité sémantique du protocole de chiffrement utilisé, l'adversaire $\mathcal{A}$ n'est pas capable de savoir si le chiffré qu'il reçoit est le chiffré de m_b ou d'un élément aléatoire. Comme l'adversaire n'obtient aucunes informations sur les messages, on peut en déduire que : $\mathsf{Adv}_2 = 0$.

En conclusion, nous pouvons majorer l'avantage de l'adversaire dans le jeu réel par :

$$\mathsf{Adv}_0 \leq \mathsf{Adv}^{\mathsf{SXDH}}_{\mathcal{GS}} + \mathsf{Adv}^{\mathsf{SXDH}}_{\mathsf{EG}}.\mathsf{Adv}^{\mathsf{SXDH}}_{\mathcal{GS}}.$$

□

Nous pouvons maintenant énoncer et prouver la résistance aux contrefaçons de notre construction.

Théorème 13. *Notre blind signature est résistante aux contrefaçons sous l'hypothèse* XDH *dans le groupe* $\mathbb{G}_1$ *et sous l'hypothèse* $\mathcal{D}_1 - \mathsf{KerMDDH}$ *dans* $\mathbb{G}_2$.

Démonstration. Afin de démonter la résistance aux contrefaçons de notre signature, nous allons adapter la preuve de notre signature SRC présentée dans la partie précédente.

En effet, si un adversaire $\mathcal{A}$ essaie de forger une signature il doit pour cela forger une SRC. Ainsi, nous simulons la réponse de l'algorithme de signature en vérifiant la validité ou non du tag renvoyé par $\mathcal{A}$. Dans la preuve de la SRC, cela concerne les jeux 0 à 2.

Ensuite nous devons savoir comment répondre si le tag est valide et différents des $\mathcal{Q}$ tags précédents. Pour cela, nous allons modifier la réponse de l'oracle de signature étape par étape. Il s'agit des jeux 3 à 4.2. Ainsi, $\mathcal{A}$ ne pourra pas distinguer si la signature qu'il reçoit est aléatoire ou valide.

L'ajout de l'algorithme de randomisation n'intervient pas dans la preuve de résistance aux contrefaçons car si la signature SRC générée par l'adversaire est aléatoire alors, en la randomisant, elle ne deviendra pas valide.

□

Conclusion

Le tableau suivant présente une comparaison des signatures déjà existantes avec nos protocoles.

Schéma	Utilisateur	Serveur	Signature	# Tours	Hypothèses
[AFG+10]	$2\mathbb{G}_1$	$3\mathbb{G}_1, 3\mathbb{G}_2$	16 $\mathbb{G}_1$, 14 $\mathbb{G}_2$	2	$q-$ADH$-$SDH
[BFPV11]	$\mathcal{O}(\ell)\mathbb{G}_1, \mathcal{O}(\ell)\mathbb{G}_2$	$2\mathbb{G}_1, 1\mathbb{G}_2$	2 $\mathbb{G}_1, 1\mathbb{G}_2$	2	SXDH
[BPV12]	$\mathcal{O}(\ell/\log(\ell))\mathbb{G}_1$	$2\mathbb{G}_1, 1\mathbb{G}_2$	2 $\mathbb{G}_1, 1\mathbb{G}_2$	2	SXDH
Ours	$2\mathbb{G}_1$	$6\mathbb{G}_1, 1\mathbb{Z}_p$	$12\mathbb{G}_1, 8\mathbb{G}_2$	2	SXDH
Asym	$(k+1)\mathbb{G}_1$	$3(k+1)\mathbb{G}_1, 1\mathbb{Z}_p$	$(k+1)(2k+4)\mathbb{G}_1$, $(k+1)(3k+3)\mathbb{G}_2$	2	$k-$MDDH
Sym	$(k+1)\mathbb{G}$	$3(k+1)\mathbb{G}, 1\mathbb{Z}_p$	$(k+1)(5k+7)\mathbb{G}$	2	$k-$MDDH

FIGURE 4.15 – Comparaison, dans le modèle standard, entre les signatures existantes et nos schémas.

Les comparaisons sont effectuées dans le modèle standard. Nous pouvons remarquer que nos schémas sont de taille constante tout en nous appuyant sur une hypothèse de sécurité standard.

Dans cette section, nous avons présenté deux résultats liés : une signature sur un chiffré ainsi qu'une signature en blanc en tour optimal.

La signature SRC, qui s'appuie sur la signature SPS de Kiltz *et al.* est effectuée en tour optimal. Elle est prouvée résistante aux contrefaçons sous une hypothèse classique *i.e.* l'hypothèse SXDH. Elle nous est nécessaire pour construire notre signature en blanc.

La signature en blanc utilise notre signature SRC ainsi que la construction de Fischlin qui permet d'obtenir des signatures en blanc en tour optimal à l'aide de NIZK.

Après avoir travaillé sur les signatures en blanc, nous pouvons présenter notre signature à base d'attributs permettant de désigner des vérifieurs.

4.3 Conclusion

Cette partie du manuscrit était concentrée sur la présentation de trois protocoles de signatures.

La première est une signature utilisant des attributs et qui permet de désigner une personne ou un groupe afin de vérifier la signature. Ce protocole, noté ABDVS, utilise différentes briques cryptographiques préexistantes tels que la signature de Waters ou les IDSPHF. Cette dernière vérifie les différentes propriétés d'une signature basée sur les attributs ainsi que celle qui désigne le vérifieur. Notre construction vérifie les propriétés suivantes : la résistance aux contrefaçons, la non-transférabilité et la *perfect-privacy* sous les hypothèses DBDH et SXDH.

Dans la seconde partie de ce chapitre, deux résultats liés étaient présentés. Le premier résultat est une signature SRC qui est de taille constante *i.e.* sa taille ne dépend pas du message. Cette signature repose sur une signature SPS (qui respecte la structure de groupe) de Kiltz *et al.*. Cette signature nous permit de construire une signature en blanc optimale en un tour et de taille constante. Cette proposition est une réponse à un problème ouvert énoncé lors de l'édition 2019 de la conférence internationale *Eurocrypt*. En utilisant la construction générale de Fischlin (à base NIZK) notre signature en blanc est optimale en un tour et ce qui est important, sa sécurité repose sur une hypothèse classique *i.e.* SXDH.

Dans le chapitre suivant, un protocole de chiffrement basé sur l'identité permettant de tracer l'identité d'un destinataire est présenté.

Chapitre 5

Assurer la traçabilité des données en utilisant des protocoles de chiffrement basés sur l'identité

Avoir la possibilité de tracer des données est une volonté de plus en plus présente. On peut, par exemple, penser à la protection des droits d'auteur d'œuvres musicales ou filmographiques. Pour ce cas précis, il existe une solution en plein développement : le *watermarking* ou le tatouage numérique. L'idée est de modifier de manière imperceptible le fichier (*e.g.* modifier un bit par octet) afin de pouvoir remonter jusqu'à l'acheteur du fichier dans le cas où son utilisation est non conforme à la législation. Notons que le tatouage numérique peut également s'appliquer aux données médicales.

Dans ce chapitre, nous n'allons pas étudier comment protéger les données d'une modification ou d'une mauvaise utilisation. Notre volonté est de protéger les utilisateurs, le système d'un utilisateur malveillant. En retirant l'anonymat de cet individu on protège les autres tout en conservant leur identité secrète. Une proposition similaire pour les signatures (signatures traçables) fut présentée par Kiayias *et al.* [KTY04].

À l'aide d'un protocole de chiffrement basé sur l'identité, nous allons montrer comment tracer l'identité du destinaire du message. Afin d'effectuer ce traçage, nous avons proposé une nouvelle primitive de protocole de chiffrement : l'*Anonymous Identity-based Encryption with Traceable identity*, noté AIBET.

Sommaire

5.1 Protocole permettant de tracer l'identité du destinataire

Dans cette partie du manuscrit, nous allons présenter une transformation effectuée sur certains chiffrements basés sur l'identité. Cette transformation permet de tracer l'identité présente dans la capsule de chiffrement *i.e.* celle du destinataire. Que signifie réellement "tracer l'identité" ? Par cette expression, nous entendons tester si l'identité du destinataire correspond à celle

recherchée ou non, et ce à l'aide d'une équation de couplages. Ce test, effectué à la fin du protocole dans un nouvel algorithme noté TVerify renvoie 1 si les deux identités (celle du destinataire et celle recherchée) sont les mêmes et 0 si ce n'est pas le cas. Pour tester l'identité, le traceur utilise une clé de traçage tsk générée par l'autorité à l'aide d'un algorithme de génération de clés TSKGen.

En résumé, en comparaison d'un chiffrement basé sur l'identité classique, nous avons ajouté :

— Une clé de traçage tsk, générée par un algorithme appelé TSKGen, qui permet de tester l'identité du destinataire sans obtenir d'information sur le message.

— Une équation de couplage, présentée dans l'algorithme TVerify, qui permet, à l'aide de tsk, de savoir si l'identité du destinataire et la même que celle recherchée.

Ce protocole, appelé *Anonymous Identity-Based Encryption with Traceable identity*, noté AIBET, peut avoir diverses applications. Tout service souhaitant vérifier les envois de ces messages sans pour autant connaître la portée du message peut être amené à l'utiliser. On peut notamment citer l'armée ou le secteur bancaire. Par exemple, pour l'armée, il serait possible pour un responsable du service de vérifier que toutes les informations transmises par ses subordonnés sont envoyées aux bons destinataires. On peut également imaginer qu'il serve de confirmation d'envoie. Un service client peut s'assurer qu'un message est parti pour le client sans en apprendre davantage.

5.1.1 Idée de la transformation

L'idée qui a permis d'aboutir à ce résultat était de connaître l'identité du destinataire d'un message chiffré. Toutefois, nous ne souhaitons pas que la personne, pour nous l'autorité, soit capable d'obtenir l'identité présente dans le chiffré puisse déchiffrer ou obtenir une quelconque information à partir de ce dernier. Dans la suite, nous allons proposer deux constructions : une basée sur l'IBE de Boyen-Waters [BW06], présentée section 3.1.3 et l'autre basée sur le *tight* IBE de Blazy *et al.* (section 3.1.2).

Afin d'éviter de laisser au traceur de l'identité un accès à trop d'informations, nous avons étudié la forme des chiffrés proposés par divers protocoles de chiffrement basés sur l'identité pour savoir s'il nous était possible d'y adjoindre les algorithmes TSKGen et TVerify.

Pour commencer, nous présentons dans la figure 5.1 une vue naïve d'un IBKEM, *i.e.* une présentation des différents algorithmes qui le composent.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}) :}$ Return mpk, msk $\underline{\mathsf{USKGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}) :}$ Return usk[id]	$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}) :}$ — $\mathsf{Enc}_1(\mathsf{mpk}, \mathsf{id})$: Return c, ρ — $\mathsf{Enc}_2(\mathsf{mpk}, \mathsf{id}, \rho)$: Return K Return K = K et c	$\underline{\mathsf{Decrypt}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, c) :}$ Return K

FIGURE 5.1 – Vue simplifiée d'IBKEM.

Dans un IBKEM, l'algorithme d'encapsulation Enc est composé de deux sous-algorithmes.

1. Enc_1 permet d'obtenir la capsule c qui va masquer la clé K en utilisant l'identité.
2. Enc_2 masque le secret dans la clé K en utilisant également l'identité.

Toutefois, tous les protocoles d'IBKEM ne sont pas construits sur le même modèle : ils n'utilisent pas tous l'identité pour générer la capsule c. Dans ce cas-là, on ne peut pas tracer l'identité sans déchiffrer le message ou au minimum donner des informations au traceur qui aurait alors la capacité de le faire. On peut notamment citer l'IBE de Boneh-Franklin [BF01] qui

est construit sur ce modèle ou encore celui de Gaborit *et al.* [GHPT17] qui repose sur les codes correcteurs d'erreurs.

Dans notre construction AIBET, nous notons par id' l'identité que l'on souhaite tracer *i.e.* nous souhaitons savoir si le message lui est destiné ou non. Tout comme dans la figure 5.1, nous présentons dans la figure 5.2 une vue simplifiée de notre construction.

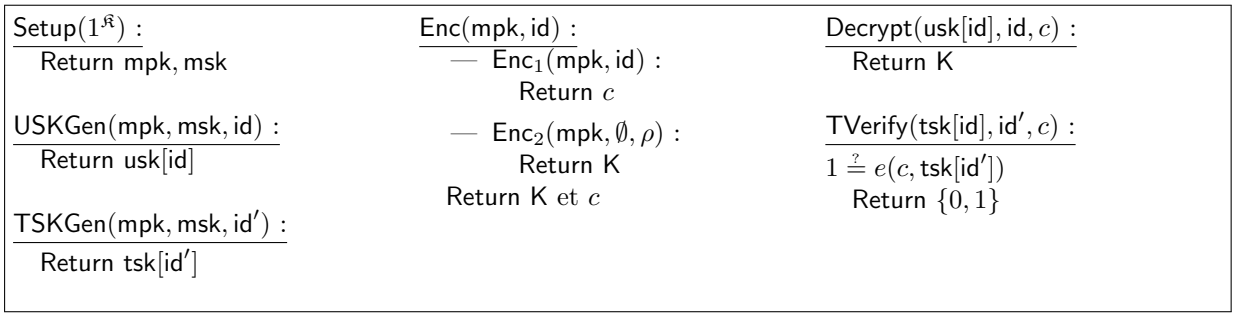


FIGURE 5.2 – Vue simplifiée d'un AIBET.

Si l'on compare les deux figures précédentes, figures 5.1 et 5.2, on peut constater que nous avons ajouté les deux algorithmes suivants :

— TSKGen est l'algorithme qui à partir de la clé maîtresse secrète msk et de l'identité que l'on souhaite tracer id' génère la clé de traçage $\mathsf{tsk}[\mathsf{id}']$. Dans les faits, cette clé est générée sur le même modèle que la clé secrète associée à l'identité de l'utilisateur notée $\mathsf{usk}[\mathsf{id}]$. Grâce à cette propriété, nous n'avons pas besoin de générer de nouveaux éléments de groupe. En effet, nous pouvons réutiliser les éléments nécessaires à la génération de $\mathsf{usk}[\mathsf{id}]$ pour générer la clé de traçage $\mathsf{tsk}[\mathsf{id}']$. Dans la suite, nous allons voir que seul le changement des variables aléatoires est nécessaire.

— TVerify est l'algorithme qui permet de savoir si l'identité utilisée lors de l'encapsulation est l'identité recherchée. Autrement dit, en reprenant les notations précédentes, si $\mathsf{id} = \mathsf{id}'$. Pour effectuer cette vérification, nous utilisons une équation de couplage qui retourne 1 seulement si les identités sont les mêmes.

Concernant la sécurité, nous attendons qu'un AIBET vérifie des hypothèses similaires à celle d'un IBE ou d'un IBKEM. Un AIBET doit vérifier l'anonymat (figure 5.3) ainsi que la sécurité sémantique (figure 5.4).

Dans le jeu d'anonymat, l'adversaire $\mathcal{A}$ ne peut pas demander de clé de traçage pour l'identité challenge. En effet, en ayant la clé de traçage pour l'identité challenge, en l'utilisant sur c^*, il pourrait savoir pour quelle identité le couple (K_b^*, c^*) est généré.

Dans le jeu de la sécurité sémantique, la présence de l'algorithme de génération de la clé de traçage TSKGen n'influe pas sur le déroulement du jeu. En effet, la clé de traçage tsk ne permettant pas de dévoiler des informations sur le message chiffré, tsk ne peut pas être utilisée à des fins malveillantes.

Nous allons maintenant voir comment nous avons pu appliquer notre construction à des protocoles de chiffrement déjà existants. Commençons par présenter notre modification basée sur l'IBE de Boyen-Waters dont nous avons rappelé la définition dans le chapitre des définitions, section 3.1.3.

5.2 Instanciations d'un AIBET

Comme la définition formelle d'un AIBET ainsi que son modèle de sécurité associé ont été présenté, nous pouvons maintenant exposer deux instanciations d'un AIBET. La première s'appuie sur l'IBE de Boyen-Waters et la seconde sur le *tight* IBE de Blazy-Kiltz-Pan.

Initialize :	$\underline{\mathsf{Enc}(\mathsf{id}^*, m^*)}$:
$(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$	//une seule requête
$b \xleftarrow{\$} \{0,1\}$	$\mathsf{K}_0^* \xleftarrow{\$} \mathcal{K}$
Return mpk	$(\mathsf{K}_1^*, c^*) \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*)$
	Return (K_b^*, c^*)
$\underline{\mathsf{USKGenO}(\mathsf{id})}$:	
$\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}\}$	$\underline{\mathsf{Finalize}(b')}$:
$\mathsf{usk}[\mathsf{id}] \xleftarrow{\$} \mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$	Return $(\mathsf{id}^* \notin \mathsf{ID}) \wedge (b = b')$
Return usk[id]	
$\underline{\mathsf{TSKGen}(\mathsf{id}')}$:	
$\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}'\}$	
$\mathsf{tsk}[\mathsf{id}'] \leftarrow \mathsf{TSKGen}(\mathsf{id}')$	
Return tsk[id']	

FIGURE 5.3 – Schéma de sécurité pour l'anonymat.

Initialize :	$\underline{\mathsf{Enc}(\mathsf{id}^*, m^*)}$:
$(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Gen}(1^{\mathfrak{K}})$	//une seule requête
$b \xleftarrow{\$} \{0,1\}$	$c_0^* \xleftarrow{\$} \mathcal{C}$
Return mpk	$c_1^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m^*)$
	Return c_b^*
$\underline{\mathsf{USKGenO}(\mathsf{id})}$:	
$\mathsf{ID} \leftarrow \mathsf{ID} \cup \{\mathsf{id}\}$	$\underline{\mathsf{Finalize}(b')}$:
$\mathsf{usk}[\mathsf{id}] \xleftarrow{\$} \mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$	Return $(\mathsf{id}^* \notin \mathsf{ID}) \wedge (b = b')$
Return usk[id]	
$\underline{\mathsf{TSKGen}(\mathsf{id}'}$:	
$\mathsf{tsk}[\mathsf{id}'] \leftarrow \mathsf{TSKGen}(\mathsf{id}')$	
Return tsk[id']	

FIGURE 5.4 – Schéma de sécurité pour la sécurité sémantique.

5.2.1 Construction basée sur Boyen-Waters

Commençons par présenter une première application de notre construction AIBET : la modification de l'IBE présenté par Boyen-Waters en 2006 dans [BW06]. Pour ce faire, nous avons ajouté nos deux algorithmes : TSKGen et TVerify. L'AIBET basé sur l'IBE de Boyen-Waters est présenté dans la figure 5.5.

Comme expliqué dans la section 5.1.1, on constate que la clé de traçage tsk, produite par TSKGen, est construite sur un modèle similaire à celui de la clé usk. Cependant, nous avons retiré la partie qui permet de déchiffrer la clé K dans l'algorithme de décapsulation Dec *i.e.* $-\omega t_2$ et $-\omega t_1$. De plus, nous avons rafraîchit les variables aléatoires r_1 et r_2 afin d'éviter des attaques visant à retrouver des éléments de clé maîtresse secrète msk. Concernant le nombre d'éléments nécessaire à la génération la clé de traçage tsk, nous constatons également ne pas avoir besoin de générer de nouveaux éléments de groupe : nous utilisons ceux présents dans les deux clés maîtresses.

Maintenant que nous avons fini d'introduire notre construction AIBET nous devons nous assurer qu'elle préserve les propriétés de sécurité de la construction de Boyen-Waters *i.e.* l'ano-

$\underline{\mathsf{Setup}(1^{\mathfrak{K}})}$:
$f, h, \omega, t_1, \ldots, t_4 \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{msk} = \omega, t_1, \ldots, t_4$
$\mathsf{mpk} = \Omega = [\omega t_1 t_2]_T, [f], [h], [t_1], \ldots, [t_4]$
Return mpk

$\underline{\mathsf{USKGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})}$:
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{usk}[\mathsf{id}] = [r_1 t_1 t_2 + r_2 t_3 t_4, -\omega t_2 - r_1 t_2 (f + h\mathsf{id}), -\omega t_1 - r_1 t_1 (f + h\mathsf{id}), -r_2 t_4 (f + h\mathsf{id}), -r_2 t_3 (f + h\mathsf{id})]$
Return $\mathsf{usk}[\mathsf{id}] \in \mathbb{G}^5$

$\underline{\mathsf{TSKGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}')}$:
$z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$

Return $\mathsf{tsk}[\mathsf{id}'] \in \mathbb{G}^5$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})}$:
$s, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{K} = [s\Omega]_T$
$c = [s(f + h\mathsf{id}), (s - s_1)t_1, s_1 t_2, (s - s_2)t_3, s_2 t_4]$
Return $\mathsf{K}, c \in \mathbb{G}_T \times \mathbb{G}^4$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, c)}$:
$\mathsf{K}' = [\sum_i c_i \cdot \mathsf{usk}[\mathsf{id}]_i]_T$
Return $\mathsf{K}' \in \mathbb{G}_T$

$\underline{\mathsf{TVerify}(\mathsf{tsk}[\mathsf{id}'], \mathsf{id}', c)}$:
Vérifie quand : $[\sum_i c_i \cdot \mathsf{tsk}[\mathsf{id}']_i]_T \stackrel{?}{=} 1_T$

FIGURE 5.5 – Boyen-Waters avec l'identité du destinataire traçable.

nymat et la sécurité sémantique. Ces deux propriétés sont prouvées dans la suite du manuscrit. Concentrons-nous dans un premier temps sur la *correctness i.e.* obtenons nous les valeurs attendues en sortie de nos algorithmes.

Commençons par les calculs vérifiant la *correctness* de la clé de l'utilisateur usk :

- $[\mathsf{K}]_T = [s\omega t_1 t_2]_T$
- $[C_0 \cdot \mathsf{usk}[\mathsf{id}]_0]_T = [s(r_1 t_1 t_2 + r_2 t_3 t_4)(f + h\mathsf{id})]_T$
- $[C_1 \cdot \mathsf{usk}[\mathsf{id}]_1]_T = [(s_1 - s)t_1 t_2 \omega + (s_1 - s)(r_1 t_1 t_2)(f + h\mathsf{id})]_T$
- $[C_2 \cdot \mathsf{usk}[\mathsf{id}]_2]_T = [-s_1 t_1 t_2 \omega + (-s_1 r_1 t_1 t_2)(f + h\mathsf{id})]_T$
- $[C_3 \cdot \mathsf{usk}[\mathsf{id}]_3]_T = [(-s r_2 t_3 t_4 + s_2 r_2 t_3 t_4)(f + h\mathsf{id})]_T$
- $[C_4 \cdot \mathsf{usk}[\mathsf{id}]_4]_T = [-s_2 r_2 t_3 t_4 (f + h\mathsf{id})]_T$

Nous pouvons également remarquer que $\mathsf{K} = \mathsf{K}' = [\sum_i C_i \cdot \mathsf{usk}[\mathsf{id}]_i]_T$. Ceci prouve que usk est correcte *i.e.* elle permet de déchiffrer.

Maintenant, vérifions la *correctness* de la clé de traçage tsk :

- $[C_0 \cdot \mathsf{tsk}[\mathsf{id}]_0]_T = [s(z_1 t_1 t_2 + z_2 t_3 t_4)(f + h\mathsf{id})]_T$
- $[C_1 \cdot \mathsf{tsk}[\mathsf{id}]_1]_T = [(s_1 - s)(t_1 t_2 z_1)(f + h\mathsf{id}')]_T$
- $[C_2 \cdot \mathsf{tsk}[\mathsf{id}]_2]_T = [-s_1 t_1 t_2 z_1 (f + h\mathsf{id}')]_T$
- $[C_3 \cdot \mathsf{tsk}[\mathsf{id}]_3]_T = [(s_2 - s)t_3 t_4 z_2 (f + h\mathsf{id}')]_T$
- $[C_4 \cdot \mathsf{tsk}[\mathsf{id}]_4]_T = [-s_2 t_3 t_4 z_2 (f + h\mathsf{id}')]_T$

En effectuant la somme des termes précédents, on obtient $V = [s(\mathsf{id} - \mathsf{id}')(t_1 t_2 z_1 + t_3 t_4 z_2)]_T$ qui est égal à 1 si et seulement si $\mathsf{id} = \mathsf{id}'$. Ceci prouve que tsk est correcte *i.e.* elle permet de simplifier l'équation de couplage pour obtenir 1 si $\mathsf{id} = \mathsf{id}'$.

Les calculs précédents permettent d'assurer la *correctness* de notre construction. Nous pouvons maintenant prouver l'anonymat et la sécurité sémantique.

Preuves de sécurité

Nous pouvons maintenant présenter les preuves de sécurité de notre construction. Le but de ces derniers est de prouver que l'on conserve les propriétés (anonymat et sécurité sémantique) de la construction de Boyen-Waters. Dans cet objectif, nous effectuons des jeux successifs nous permettant de nous ramener à la preuve originale afin de montrer que l'ajout de nos deux algorithmes n'influe pas sur la sécurité du protocole. Comme Boyen-Waters, nous allons prouver que notre construction est anonyme sous l'hypothèse DLin (section 2.1.1) et vérifie la sécurité sémantique sous l'hypothèse DBDH (section 2.1.1).

Anonymat Afin de prouver l'anonymat, nous allons nous ramener à un challenge associé à l'hypothèse de sécurité DLin. Le but est de montrer que ce challenge, qui se présente sous la forme d'un tuple de valeurs, est indiscernable d'un tuple contenant une valeur aléatoire. Pour ce faire, tout comme dans la preuve de Boyen-Waters, nous allons procéder via des jeux de sécurité.

Théorème 14. *Soit $\mathbb{G}$ un groupe bilinéaire. Supposons que (t,ϵ)-*DLin *soit valide dans $\mathbb{G}$. Il n'existe pas d'adversaire $\mathcal{A}$ en temps t capable de distinguer deux identités avec un avantage plus grand que ϵ.*

Démonstration. Pour cette preuve, il est nécessaire de noter que nous n'autorisons pas l'adversaire à demander une clé de traçage pour l'identité challenge id^* *i.e.* $\mathsf{tsk}[\mathsf{id}^*]$. Nous notons par $\mathcal{Q}_{\mathsf{ID}}$ l'ensemble des identités qui ont fait l'objet d'une requête par l'adversaire.

Le challenger reçoit un tuple DLin de la forme : $[z_1, z_2, z_1z_3, z_2z_4, Z]$ où Z est soit $[z_3 + z_4]$ soit un élément aléatoire de $\mathbb{G}$. Les différents jeux utilisés dans cette preuve sont présentés dans les figures 5.6, 5.7 et 5.8. La preuve se découpe en quatre jeux distincts en considérant que le jeu $\mathbf{G_0}$ est le jeu initial *i.e.* aucune modification n'est apportée aux paramètres de l'AIBET.

— Dans le premier jeu, noté $\mathbf{G_1}$, par rapport au jeu $\mathbf{G_0}$, nous modifions seulement les valeurs aléatoires utilisées pour la génération de usk par USKGen. Par construction, nous modifions également les valeurs aléatoires utilisées dans TSKGen.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}, \mathsf{id}^*)}$:
$y, z_1, z_2, t_1, \ldots, t_4 \xleftarrow{\$} \mathbb{Z}_p$
Posons : $\omega = z_1z_2$, $[f] = [y - z_1\mathsf{id}^*]$ et $[h] = [z_1]$
Return $\mathsf{mpk} = (\Omega = [\omega]_T, [t_1], \ldots, [t_4])$

$\underline{\mathsf{Guess}(\beta)}$:
Return $(\mathsf{id}^* \notin \mathcal{Q}_{\mathsf{ID}}) \wedge \beta$

$\underline{\mathsf{USKGen}(\mathsf{id})}$:
$\mathcal{Q}_{\mathsf{ID}} = \mathcal{Q}_{\mathsf{ID}} \cup \{\mathsf{id}\}$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
$\tilde{r}_1 = r_1 - \frac{z_2}{\mathsf{id}-\mathsf{id}^*}$ et $\tilde{r}_2 = r_2$

$\mathsf{usk}[\mathsf{id}] = [\tilde{r}_1t_1t_2 + \tilde{r}_2t_3t_4, -\omega t_2 - \tilde{r}_1t_2(f + h\mathsf{id}), -\omega t_1 - \tilde{r}_1t_1(f + h\mathsf{id}), -\tilde{r}_2t_4(f + h\mathsf{id}), -\tilde{r}_2t_3(f + h\mathsf{id})]$
Return $\mathsf{usk}[\mathsf{id}]$

$\underline{\mathsf{TSKGen}(\mathsf{id})}$:
$\mathcal{Q}_{\mathsf{ID}} = \mathcal{Q}_{\mathsf{ID}} \cup \{\mathsf{id}\}$
$w_1, w_2 \xleftarrow{\$} \mathbb{Z}_p$
$\tilde{w}_1 = w_1 - \frac{z_2}{\mathsf{id}-\mathsf{id}^*}$ et $\tilde{w}_2 = w_2$

$\mathsf{tsk}[\mathsf{id}] = [\tilde{w}_1t_1t_2 + \tilde{w}_2t_3t_4, -\tilde{w}_1t_2(f + h\mathsf{id}), -\tilde{w}_1t_1(f + h\mathsf{id}), -\tilde{w}_2t_4(f + h\mathsf{id}), -\tilde{w}_2t_3(f + h\mathsf{id})]$
Return $\mathsf{tsk}[\mathsf{id}]$

$\underline{\mathsf{Enc}(\mathsf{id}^*)}$:
$s, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$
$Y = t_1(s - s_1), Y' = t_3(s - s_2)$,
$c = [sy, Y, t_2s_1, Y', t_4s_2]$
Return c

FIGURE 5.6 – Étapes du jeu $\mathbf{G_1}$ utilisé dans la preuve de l'anonymat.

— Dans le deuxième jeu, noté $\mathbf{G_2}$, nous allons une nouvelle fois modifier les éléments aléatoires $\tilde{r_1}$ et $\tilde{r_2}$ du jeu $\mathbf{G_1}$ afin de se rapprocher d'une distribution aléatoire. Les modifications effectuées sont rapportées dans la figure 5.7.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}, \mathsf{id}^*) :}$
$y, z_1, z_2, t_1, \ldots, t_4 \xleftarrow{\$} \mathbb{Z}_p$
Posons : $[f] = [y - z_1\mathsf{id}^*]$ et $[h] = [z_1]$
$\alpha, \omega \xleftarrow{\$} \mathbb{Z}_p$
Posons $[f] = [y - z_2\mathsf{id}^*\alpha]$, $[h] = [z_2\alpha]$
et $t_1 = z_2, t_2 = z_1$

Return $\mathsf{mpk} = (\Omega = [\omega]_T, [t_1], \ldots, [t_4])$

$\underline{\mathsf{Guess}(\beta) :}$
Return $(\mathsf{id}^* \notin \mathcal{Q}_{\mathsf{ID}}) \wedge \beta$

$\underline{\mathsf{USKGen}(\mathsf{id}) :}$
$\mathcal{Q}_{\mathsf{ID}} = \mathcal{Q}_{\mathsf{ID}} \cup \{\mathsf{id}\}$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$

$\tilde{r}_1 = \frac{r_1\alpha(\mathsf{id}-\mathsf{id}^*)}{\alpha(\mathsf{id}-\mathsf{id}^*)z_2+y}$ et $\tilde{r}_2 = r_2 + \frac{yz_1r_1}{t_3t_4(\alpha(\mathsf{id}-\mathsf{id}^*)z_2+y)}$

$\mathsf{usk}[\mathsf{id}] = [\tilde{r}_1t_1t_2 + \tilde{r}_2t_3t_4, -\omega t_2 - \tilde{r}_1t_2(f + h\mathsf{id}),$
$-\omega t_1 - \tilde{r}_1t_1(f + h\mathsf{id}),\ -\tilde{r}_2t_4(f + h\mathsf{id}), -\tilde{r}_2t_3(f + h\mathsf{id})]$
Return $\mathsf{usk}[\mathsf{id}]$

$\underline{\mathsf{TSKGen}(\mathsf{id}) :}$
$\mathcal{Q}_{\mathsf{ID}} = \mathcal{Q}_{\mathsf{ID}} \cup \{\mathsf{id}\}$
$w_1, w_2 \xleftarrow{\$} \mathbb{Z}_p$

$\tilde{w}_1 = \frac{w_1\alpha(\mathsf{id}-\mathsf{id}^*)}{\alpha(\mathsf{id}-\mathsf{id}^*)z_2+y}$
$\tilde{w}_2 = w_2 + \frac{yz_1r_1}{t_3t_4(\alpha(\mathsf{id}-\mathsf{id}^*)z_2+y)}$
$\mathsf{tsk}[\mathsf{id}] = [\tilde{w}_1t_1t_2 + \tilde{w}_2t_3t_4, -\tilde{w}_1t_2(f + h\mathsf{id}), -\tilde{w}_1t_1(f + h\mathsf{id}), -\tilde{w}_2t_4(f + h\mathsf{id}), -\tilde{w}_2t_3(f + h\mathsf{id})]$
Return $\mathsf{tsk}[\mathsf{id}]$

$\underline{\mathsf{Enc}(\mathsf{id}^*) :}$
$s, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$
$Y' = t_3(s - s_2)$,
$Y \xleftarrow{\$} \mathbb{G}$, $c = [sy, Y, t_2s_1, Y', t_4s_2]$
Return c

FIGURE 5.7 – Étapes du jeu $\mathbf{G_2}$ utilisé dans la preuve de l'anonymat.

— Dans le troisième et dernier jeu, noté $\mathbf{G_3}$, nous ne modifions ni USKGen ni TSKGen par rapport au jeu précédent comme nous pouvons le voir dans la figure 5.8. Dans ce jeu, nous rendons totalement aléatoires les valeurs renvoyées par Enc. Ainsi, nous obtenons un challenge DLin. Par conséquent, si l'adversaire gagne le jeu, cela signifie qu'il a cassé cette hypothèse en temps t, or cela n'est possible qu'avec une probabilité ϵ-négligeable.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}, \mathsf{id}^*) :}$
$y, z_1, z_2, t_1, \ldots, t_4 \xleftarrow{\$} \mathbb{Z}_p$
Posons $\omega = z_1z_2$, $[f] = [y - z_1\mathsf{id}^*]$ et $[h] = [z_1]$
$\alpha, \omega \xleftarrow{\$} \mathbb{Z}_p$
Posons $[f] = [y - z_2\mathsf{id}^*\alpha]$, $[h] = [z_2\alpha]$,
$[t_3] = [z_2]$ et $[t_4] = [z_1]$
Return $\mathsf{mpk} = (\Omega = [\omega]_T, [t_1], \ldots, [t_4])$

$\underline{\mathsf{Guess}(\beta) :}$
Return $(\mathsf{id}^* \notin \mathcal{Q}_{\mathsf{ID}}) \wedge \beta$

$\underline{\mathsf{USKGen}(\mathsf{id}) :}$
Identique au jeu $\mathbf{G_2}$
Return $\mathsf{usk}[\mathsf{id}]$

$\underline{\mathsf{TSKGen}(\mathsf{id}) :}$
Identique au jeu $\mathbf{G_2}$
Return $\mathsf{tsk}[\mathsf{id}]$

$\underline{\mathsf{Enc}(\mathsf{id}^*) :}$
$s, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{K} \xleftarrow{\$} \mathbb{G}_T$
$Y, Y' \xleftarrow{\$} \mathbb{G}$,
$c = [sy, Y, t_2s_1, Y', t_4s_2]$
Return c

FIGURE 5.8 – Étapes du jeu $\mathbf{G_3}$ utilisé dans la preuve de l'anonymat.

Les jeux précédents simulent successivement l'AIBET et comme le jeu $\mathbf{G_3}$ prouve que l'adversaire ne peut gagner le jeu que s'il est apte à résoudre le problème DLin. Par conséquent, notre AIBET est anonyme sous l'hypothèse DLin. □

Sécurité Sémantique

Sur le même modèle que la preuve sur l'anonymat, nous allons montrer que notre AIBET vérifie toujours la sécurité sémantique en suivant le modèle de la preuve de Boyen-Waters.

Théorème 15. *Soit $\mathbb{G}$ un groupe bilinéaire. Supposons que l'hypothèse décisionnel de (t, ϵ)-DBDH soit valide dans $\mathbb{G}$. Il n'existe pas d'adversaire $\mathcal{A}$ en temps t capable de distinguer deux chiffrés avec un avantage plus grand que ϵ.*

En comparaison de la preuve sur l'anonymat, pour la sécurité sémantique, nous autorisons l'adversaire à effectuer des requêtes à TSKGen sur l'identité challenge id^*. Ainsi, par rapport à la preuve précédente, nous modifions seulement l'algorithme Enc.

Démonstration. Afin de prouver que notre construction est toujours sémantiquement sûre, nous devons construire un simulateur qui joue le challenge DBDH. Pour cela, il reçoit $[z_1, z_2, z_3], [Z]_T$ où Z est soit $[z_1 z_2 z_3]_T$ soit un élément aléatoire de $\mathbb{G}_T$. La probabilité d'obtenir l'un ou l'autre est identique *i.e.* elle vaut ϵ. L'objectif de la preuve est de montrer que les deux cas sont indiscernables. Cette preuve est composée de deux jeux de sécurité.

— $\mathbf{G_0}$ est le protocole original n'ayant subi aucune modification.

— Dans $\mathbf{G_1}$, il s'agit du jeu où l'on introduit $[Z]_T$. On commence par remplacer r_1 et et r_2 dans USKGen par $\tilde{r_1}$ et $\tilde{r_2}$ qui sont sous la même forme que ceux utilisés dans la preuve d'anonymat (figure 5.7). Comme on a pu le voir précédemment, la modification de r_1 et r_2 n'influe pas sur l'algorithme TSKGen. On remplace également $[z_1 z_2 z_3]_T$ par $[Z]_T$ dans l'algorithme Enc.

Ainsi, l'adversaire joue, avec une probabilité ϵ le jeu $\mathbf{G_0}$ ou le jeu $\mathbf{G_1}$. Les algorithmes TSKGen et TVerify que nous avons ajouté n'ont pas d'effet sur les modifications effectuées sur le chiffré. Par conséquent, on peut appliquer directement la preuve de Boyen-Waters. Cela prouve la sécurité sémantique de notre AIBET. □

Ces différentes preuves nous permettent de conclure que nous conservons les propriétés de sécurité tout en ajoutant nos protocoles de traçage.

Suite à cette construction, nous présentons un AIBET, basé sur le *tight* IBE de Blazy-Kiltz-Pan, qui, comme précédemment, conserve les propriétés de l'IBE utilisé.

5.2.2 Construction basée sur Blazy-Kiltz-Pan

Avant de présenter notre construction, nous devons introduire des MAC affines spéciaux. Ici, nous avons besoin de MAC qui utilisent les *Smooth Projective Hash Function* (ces notions furent présentées dans la section 2.1.3). Ces MAC, notés $\mathsf{MAC_{SPHF}}$, respectent la sécurité sémantique (RPR-CMA) dont la définition est présentée dans la section 3.1.2. Nous les utilisons afin de maintenir la sécurité de notre AIBET reposant sur l'IBE de Blazy-Kiltz-Pan.

Préliminaires

Dans le chapitre des définitions, section 2.1.3, nous avons présenté les MAC affines ainsi que les MAC affines partiels. Les $\mathsf{MAC_{SPHF}}$ sont définis dans la figure 5.9 où un message $\mathbf{m}$ est un élément de $\mathcal{M} = \mathbb{Z}_p^l$

$\underline{\mathsf{Gen}_{\mathsf{MAC}}(\mathfrak{K})}$:	$\underline{\mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, m)}$:	$\underline{\mathsf{Ver}(\mathsf{sk}_{\mathsf{MAC}}, \tau, m)}$:
$\mathbf{B} \xleftarrow{\$} \mathcal{D}_k$	$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k$, $\mathbf{t} = \mathbf{Bs} \in \mathbb{Z}_p^{k+1}$	Si $u = (\mathbf{x}_0^\top + \sum_{i=1}^{\lvert m\rvert} m_i \cdot \mathbf{x}_i^\top)\mathbf{t} + x'$
$\mathbf{x}_0, \ldots, \mathbf{x}_\ell \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, $x' \xleftarrow{\$} \mathbb{Z}_p$	$u = (\mathbf{x}_0^\top + \sum_{i=1}^{\lvert m\rvert} m_i \cdot \mathbf{x}_i^\top)\mathbf{t} + x' \in \mathbb{Z}_p$	Return 1
Return $\mathsf{sk}_{\mathsf{MAC}} = (\mathbf{B}, \mathbf{x}_0, \ldots, \mathbf{x}_\ell, x')$	Return $\tau = ([\mathbf{t}]_2, [u]_2) \in \mathbb{G}_2^{k+1} \times \mathbb{G}_2$	Sinon Return 0

FIGURE 5.9 – Définition des $\mathsf{MAC}_{\mathsf{SPHF}}$.

Théorème 16. *Les* $\mathsf{MAC}_{\mathsf{SPHF}}$ *sont* RPR-CMA^0-secure *sous l'hypothèse* $\mathcal{D}_k$-MDDH. *En particuliers, pour tout adversaire* $\mathcal{A}$*, il existe un adversaire* $\mathcal{D}$ *avec* $\mathcal{T}(\mathcal{A}) \approx \mathcal{T}(\mathcal{D})$ *et* $\mathsf{Adv}^{\mathsf{rpr\text{-}cma0}}_{\mathsf{MAC}_{\mathsf{SPHF}}}(\mathcal{A}) \leq 2Q(\mathsf{Adv}_{\mathcal{D}_k,\mathsf{ggen}}(\mathcal{D}) + 1/p)$*, où* Q *est le nombre totale de requêtes à* $\mathsf{Eval}(\cdot)$ *et à* $\mathsf{PEval}(\cdot)$.

Dans le but de prouver ce théorème, nous allons construire une preuve sur le modèle des jeux présentés dans la section 3.1.2 du chapitre des définitions. Soit $\mathcal{A}$ un adversaire contre la sécurité RPR-CMA des $\mathsf{MAC}_{\mathsf{SPHF}}$. Nous définissons une suite de jeux de sécurité comme dans la figure 5.10. Ces derniers seront utilisés dans différents lemmes qui permettront d'aboutir à la démonstration du théorème 16.

$\underline{\mathsf{Initialize}}$: // Jeux $\mathbf{G_0} - \mathbf{G_2}$	$\underline{\mathsf{Eval}(m)}$: // Jeux $\mathbf{G_0}$
$\mathbf{B} \xleftarrow{\$} \mathcal{D}_k$	$\mathcal{Q}_{\mathcal{M}} = \mathcal{Q}_{\mathcal{M}} \cup \{m\}$
$\forall j \in [\![1, l]\!] : \mathbf{x}_{j,0}, \mathbf{x}_{j,1} \leftarrow \mathbb{Z}_p^{k+1}$	$s \xleftarrow{\$} \mathbb{Z}_p^k$, $\mathbf{t} = \mathbf{B}s \in \mathbb{Z}_p^{k+1}$
Return $[\mathbf{B}]_2, ([\mathbf{x}_{j,b}^\top \mathbf{B}]_2)_j$	$u = \sum_{j=1}^{l} \mathbf{m}_j \mathbf{x}_j^\top \mathbf{t} + x'$
	Return $([\mathbf{t}]_2, [u]_2)$
$\underline{\mathsf{Chal}(m^*)}$: // Jeux $\mathbf{G_0} - \mathbf{G_{1,\mathcal{Q}+1}}$ $\boxed{\mathbf{G_2}}$	$\underline{\mathsf{Eval}(m)}$: // Jeux $\mathbf{G_{1,i}}$, $\boxed{\mathbf{G'_{1,i}}}$
$h \xleftarrow{\$} \mathbb{Z}_p$;	$\mathcal{Q}_{\mathcal{M}} = \mathcal{Q}_{\mathcal{M}} \cup \{m\}$
$\mathbf{h}_0 = (\sum_{j=1}^{l} \mathbf{m}_j^* \mathbf{x}_j) \cdot h$; $h_1 = x' \cdot h \in \mathbb{Z}_p$	$(1 \leq c \leq Q)$
$\boxed{\mathbf{h}_0 \xleftarrow{\$} \mathbb{Z}_p^{k+1} ; h_1 \xleftarrow{\$} \mathbb{Z}_p}$	Si $c < i$ alors $(\mathbf{t}, u) \xleftarrow{\$} \mathbb{Z}_p^{k+1} \times \mathbb{Z}_p$
Return $([h]_1, [\mathbf{h}_0]_1, [h_1]_T)$	Si $c > i$ alors $s \xleftarrow{\$} \mathbb{Z}_p^k$,
$\underline{\mathsf{Eval}(m)}$: // $\mathbf{G_2}$	$\boxed{\mathbf{t} = \mathbf{B}s \in \mathbb{Z}_p^{k+1}}$
$\mathcal{Q}_{\mathcal{M}} = \mathcal{Q}_{\mathcal{M}} \cup \{m\}$	$u = \sum_{j=1}^{l} \mathbf{m}_j \mathbf{x}_j^\top \mathbf{t} + x'$
$(\mathbf{t}, u) \xleftarrow{\$} \mathbb{Z}_p^{k+1} \times \mathbb{Z}_p$	Return $([\mathbf{t}]_2, [u]_2)$
Return $([\mathbf{t}]_2, [u]_2)$	$\underline{\mathsf{Finalize}(d \in \{0,1\})}$: // Jeux $\mathbf{G_0} - \mathbf{G_2}$
	Return $d \wedge (m^* \notin \mathcal{Q}_{\mathcal{M}})$

FIGURE 5.10 – Jeux $\mathbf{G_0}$, $(\mathbf{G_{1,i}}, \mathbf{G'_{1,i}})_{(0 \leq i \leq m)}$ et $\mathbf{G_2}$ pour les preuves des lemmes 2 à 4.

Lemme 2. *Il existe un adversaire* $\mathcal{B}$ *avec* $\mathcal{T}(\mathcal{B}) \approx \mathcal{T}(\mathcal{A})$ *et* $\mathsf{Adv}_{\mathcal{D}_k,\mathsf{ggen}}(\mathcal{B}) \geq |\Pr[\mathbf{G'^{\mathcal{A}}_{1,i}} \Rightarrow 1] - \Pr[\mathbf{G^{\mathcal{A}}_{1,i}} \Rightarrow 1]|$

Démonstration. Les jeux $\mathbf{G_{1,i}}$ et $\mathbf{G'_{1,i}}$ sont seulement différents par la distribution de $\mathbf{t}$ renvoyé par l'oracle Eval pour ses i-ème requêtes, c'est-à-dire, $\mathbf{t} \in \mathit{span}(\mathbf{B})$ où il est uniforme. Grâce à cela, nous obtenons une réduction simple de l'hypothèse $\mathcal{D}_k$-MDDH. □

Lemme 3. $|\Pr[\mathbf{G_{1,i+1}}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G^{\mathcal{A}}_{1,i}} \Rightarrow 1]| \leq 1/p$.

Démonstration. Soit $\mathbf{m}$ la i-ième requête à Eval et soit $([\mathbf{t}]_2, [u]_2)$ le tag. Comme $\mathbf{m} \neq \mathbf{m}^*$, il existe un indice i' tels que $\mathbf{m}_{i'} \neq \mathbf{m}^*_{i'}$, où $\mathbf{m}_{i'}$ (resp. $\mathbf{m}^*_{i'}$) correpond à l'i-ième entrée de $\mathbf{m}$ (resp.$\mathbf{m}^*$). On assume que x' et $\mathbf{x}_j (j \notin \{0, i'\})$ soient connus par $\mathcal{A}$. Du point de vue de la théorie

de l'information, l'adversaire $\mathcal{A}$ doit aussi connaître $\mathbf{B}^\top \mathbf{x}_0$ et $\mathbf{B}^\top \mathbf{x}_{i'}$ par la c-ième requêtes avec $c > i$. Ainsi, $\mathcal{A}$ obtient théoriquement les équations suivantes avec les variables inconnues $\begin{pmatrix}\mathbf{x}_0\\ \mathbf{x}_{i'}\end{pmatrix}$:

$$M = \begin{pmatrix}\mathbf{B}^\top \mathbf{x}_0\\ \mathbf{B}^\top \mathbf{x}_{i'}\\ \mathbf{h}_0\\ u - x'\end{pmatrix} = \begin{pmatrix}\mathbf{B}^\top & 0\\ 0 & \mathbf{B}^\top\\ h \cdot \mathbf{I}_{k+1} & \mathbf{m}_i^* h \cdot \mathbf{I}_{k+1}\\ \mathbf{t}^\top & \mathbf{m}_{i'}\mathbf{t}^\top\end{pmatrix} \cdot \begin{pmatrix}\mathbf{x}_0\\ \mathbf{x}_{i'}\end{pmatrix}$$

où $\mathbf{I}_{k+1}$ est la matrice identité de taille $(k+1)\times(k+1)$. Pour montrer que $u - x'$ est linéairement indépendant de $\mathbf{B}^\top \mathbf{x}_0$, $\mathbf{B}^\top \mathbf{x}_{i'}$ et $\mathbf{h}_0$, nous utilisons le fait que la dernière ligne de la matrice M est linéairement indépendante de toutes les autres lignes. Comme $\mathbf{t} \notin span(\mathbf{B})$ (sauf avec probabilité $1/p$), il est indépendant de la matrice $\mathbf{B}^\top$. Grâce à $\mathbf{m}_{i'} \neq \mathbf{m}_{i'}^*$, la dernière ligne de M est indépendante des lignes $2k+1$ à $3k+1$. Ainsi, du point de vue de l'adversaire, u est uniformément aléatoire. □

Lemme 4. $\Pr[\mathbf{G}_{\mathbf{2}}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathbf{G}_{\mathbf{1},\mathcal{Q}+\mathbf{1}}^{\mathcal{A}} \Rightarrow 1]$

Démonstration. L'adversaire $\mathcal{A}$ peut faire au plus $\mathcal{Q}$ requêtes à l'oracle Eval. Dans chaque jeu $\mathbf{G}_{\mathbf{1},\mathcal{Q}+\mathbf{1}}$ et dans $\mathbf{G}_{\mathbf{2}}$, toutes les réponses de Eval sont uniformément aléatoires et indépendantes de la clé secrète $\mathsf{sk}_{\mathsf{MAC}}$. Par conséquent, les valeurs $\mathbf{h}_0$ et h_1 de $\mathbf{G}_{\mathbf{1},\mathcal{Q}+\mathbf{1}}$ sont uniformes du point de vue de $\mathcal{A}$. □

Lemme 5. *Il existe un adversaire* $\mathcal{B}$ *avec* $\mathsf{TIME}(\mathcal{B}) \approx \mathsf{TIME}(\mathcal{A})$ *et* $\mathcal{Q}(\mathbf{Adv}_{\mathcal{D}_k,\mathsf{ggen}}(\mathcal{B}) + 1/p) \geq |\Pr[\mathsf{RPR}-\mathsf{CMA}_{\mathsf{rand}}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_{\mathbf{2}}^{\mathcal{A}} \Rightarrow 1]|$.

Démonstration. Les étapes de la preuve de ce lemme sont présentées sous forme de jeux dans la figure 5.11. □

Initialize : // Jeux $\mathbf{H_0}0 - \mathbf{H_2}$
$\mathbf{B} \xleftarrow{\$} \mathcal{D}_k$; $x'_0 \xleftarrow{\$} \mathbb{Z}_p$
Pour $i = 1, \ldots, \ell$: $\mathbf{x}_i \leftarrow \mathbb{Z}_p^{k+1}$
Return $([\mathbf{B}]_2, ([\mathbf{x}_{j,b}^\top \mathbf{B}]_2)_j)$
 	
Chal(m^*) : //Jeux $\mathbf{H_0}$ - H_2
$h \xleftarrow{\$} \mathbb{Z}_p$ $\mathbf{h}_0 \xleftarrow{\$} \mathbb{Z}_p^{k+1}$; $h_1 \xleftarrow{\$} \mathbb{Z}_p$
Return $([h]_1, [\mathbf{h}_0]_1, [h_1]_T)$
 	
Finalize($d \in \{0,1\}$) : // Jeux $\mathbf{H_0} - \mathbf{H_2}$
Return $d \wedge (\mathbf{m}^* \notin \mathcal{Q}_\mathcal{M})$
 	
Eval(m) : // Jeux $\mathbf{H_0}$
$\mathcal{Q}_\mathcal{M} = \mathcal{Q}_\mathcal{M} \cup \{m\}$
$(\mathbf{t}, u) \xleftarrow{\$} \mathbb{Z}_q^{k+1} \times \mathbb{Z}_p$
Return $([\mathbf{t}]_2, [u]_2)$
	Eval(m) : // Jeux $\mathbf{H_{1,i}}$, $\boxed{\mathbf{H'_{1,i}}}$
	$\mathcal{Q}_\mathcal{M} = \mathcal{Q}_\mathcal{M} \cup \{m\}$
	$(1 \leq c \leq Q)$
	Si $c > i$, alors $(\mathbf{t}, u) \xleftarrow{\$} \mathbb{Z}_q^{k+1} \times \mathbb{Z}_p$
	Si $c < i$ alors $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^k, \mathbf{t} = \mathbf{Bs}$;
	$u = (\mathbf{x}_0^\top + \sum_{j=1}^{\ell} \mathbf{m}_j \cdot \mathbf{x}_j^\top)\mathbf{t} + x'$
	
	Si $c = i$ alors
	$\mathbf{t} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$
	$\boxed{\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k, \quad \mathbf{t} = \mathbf{Bs}}$
	$u = (\mathbf{x}_0^\top + \sum_{j=1}^{\ell} \mathbf{m}_j \cdot \mathbf{x}_j^\top)\mathbf{t} + x'$
	
	Return $([\mathbf{t}]_2, [u]_2)$
	
	Finalize($d \in \{0,1\}$) : // Jeux $\mathbf{H_0} - \mathbf{H_2}$
	Return $d \wedge (m^* \notin \mathcal{Q}_\mathcal{M})$

FIGURE 5.11 – Jeux $\mathbf{H_0}$, $(\mathbf{H_{1,i}}, \mathbf{H'_{1,i}})_{(0 \leq i \leq \mathcal{Q})}$, $\mathbf{H_{1,Q+1}}$ et $\mathbf{H_2}$ pour la preuve du lemme 5.

Démonstration. Théorème 16 Ces différents lemmes permettent de conclure la preuve du théorème 16. □

Nous allons maintenant montrer que les $\mathsf{MAC}_{\mathsf{SPHF}}$ vérifient le théorème suivant.

Théorème 17. $\mathsf{MAC}_{\mathsf{SPHF}}$ *sont* $\mathsf{RPR\text{-}CMA}^1$*-secure sous l'hypothèse* $\mathcal{D}_k$*-*MDDH*. En particuliers, pour tout adversaire* $\mathcal{A}$*, il existe un adversaire* $\mathcal{D}$ *avec* $\mathcal{T}(\mathcal{A}) \approx \mathcal{T}(\mathcal{D})$ *et* $\mathsf{Adv}^{\mathsf{rpr\text{-}cma1}}_{\mathsf{MAC}_{\mathsf{SPHF}}}(\mathcal{A}) \leq 2Q_P(Q_E+Q_P)(\mathbf{Adv}_{\mathcal{D}_k,\mathsf{ggen}}(\mathcal{D})+1/p)$*, où* Q_E *est le nombre totale de requêtes à* $\mathsf{Eval}(\cdot)$ *et* Q_P *le nombre de requêtes à* $\mathsf{PEval}(\cdot)$*.*

Démonstration. Le fonctionnement de cette preuve est similaire à la précédente. Toutefois, cette fois-ci, nous devons pouvoir répondre à une requête de $\mathsf{PEval}(\mathbf{m}^*)$, avec $\mathbf{m}^*$ le message challenge.

L'adversaire devra deviner si la réponse de PEval à sa requête est valide ou non d'où l'ajout du coefficient Q_P dans l'inéquation. □

Maintenant que nous avons fini de présenter les $\mathsf{MAC}_{\mathsf{SPHF}}$ ainsi que leurs propriétés de sécurité, nous pouvons exposer la modification du protocole d'IBKEM de Blazy-Kiltz-Pan.

AIBET basé sur un *tight* IBE

La deuxième construction est basée sur l'IBKEM présenté en 2014 par Blazy *et al.* dans [BKP14]. De même que pour la construction précédente, nous avons ajouté les deux algorithmes TSKGen et TVerify. La clé de traçage tsk est générée sur le même modèle que la clé de l'utilisateur usk. Afin de construire tsk, nous avons retiré la partie qui permettait d'accéder au secret *i.e.* le vecteur $\mathbf{y}'$. La figure 5.12 présente cette construction, où l'on note par f une fonction définie publiquement tel que $f_i : \mathcal{M} \to \mathbb{Z}_p$.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}})}$:
$\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$
$\mathsf{sk}_{\mathsf{MAC}} = (\mathbf{B}, \mathbf{x}_0, \ldots, \mathbf{x}_\ell, x') \xleftarrow{\$} \mathsf{Gen}_{\mathsf{MAC}}(\mathfrak{K})$
$\forall i \in [\![0,\ell]\!] : \mathbf{Y}_i \xleftarrow{\$} \mathbb{Z}_p^{k\times n}$,
$\mathbf{Z}_i = (\mathbf{Y}_i^\top | \mathbf{x}_i) \cdot \mathbf{A} \in \mathbb{Z}_p^{n\times k}$
$\mathbf{y'} \xleftarrow{\$} \mathbb{Z}_p^k$; $\mathbf{z'} = (\mathbf{y'}^\top | x') \cdot \mathbf{A} \in \mathbb{Z}_p^{1\times k}$
$\mathsf{mpk} = (\mathcal{G}, [\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{[\![0,\ell]\!]}, [\mathbf{z'}]_1)$,
$\mathsf{msk} = (\mathsf{sk}_{\mathsf{MAC}}, (\mathbf{Y}_i)_{[\![0,\ell]\!]}, \mathbf{y'})$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{mpk}, \mathsf{id})}$:
$([\mathbf{t}]_2, [u]_2) \xleftarrow{\$} \mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, \mathsf{id})$
$\mathbf{v} = \sum_{i=0}^{\ell} f_i(\mathsf{id})\mathbf{Y}_i\mathbf{t} + \mathbf{y'} \in \mathbb{Z}_p^k$
Return $\mathsf{usk}[\mathsf{id}] = ([\mathbf{t}]_2, [u]_2, [\mathbf{v}]_2) \in \mathbb{G}_2^{n+1+k}$

$\underline{\mathsf{TSKGen}(\mathsf{msk}, \mathsf{mpk}, \mathsf{id}')}$:
$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$, $\mathbf{t} = \mathbf{Bs} \in \mathbb{Z}_p^{k+1}$
$\mathsf{tsk}[\mathsf{id}'] = [\mathbf{t}, (\sum_{i=0}^{l} f_i(\mathsf{id}') \begin{pmatrix}\mathbf{Y}_i \\ \mathbf{x}_i\end{pmatrix} \cdot \mathbf{t}]_2$
Return $\mathsf{tsk}[\mathsf{id}'] \in \mathbb{G}_2^{n+k+1}$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})}$:
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$
$\mathbf{c}_0 = \mathbf{Ar} \in \mathbb{Z}_p^{k+1}$,
$\mathbf{c}_1 = (\sum_{i=0}^{l} f_i(\mathsf{id})\mathbf{Z}_i) \cdot \mathbf{r} \in \mathbb{Z}_p^n$
$\mathsf{K} = \mathbf{z'} \cdot \mathbf{r} \in \mathbb{Z}_p$
Return $\mathsf{K} = [\mathsf{K}]_T$ et $c = ([\mathbf{c}_0]_1, [\mathbf{c}_1]_1) \in \mathbb{G}_1^{n+k+1}$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, c)}$:
$\mathsf{K} = e\left([\mathbf{c}_0]_1, \begin{bmatrix}\mathbf{v} \\ u\end{bmatrix}_2\right) / e([\mathbf{c}_1]_1, [\mathbf{t}]_2)$
Return $\mathsf{K} \in \mathbb{G}_T$

$\underline{\mathsf{TVerify}(\mathsf{tsk}[\mathsf{id}'], \mathsf{id}', c)}$:
$1_T \stackrel{?}{=} e([\mathbf{c}_1]_1, [\mathsf{tsk}_1^\top]_2) / e([\mathbf{c}_0]_1, [\mathsf{tsk}_2^\top]_2)$

FIGURE 5.12 – AIBET utilisant le protocole de Blazy-Kiltz-Pan

De même que pour la construction basée sur l'IBE de Boyen-Waters, nous avons construit l'algorithme TSKGen sur le modèle de l'USKGen original. En termes d'éléments de groupe, nous utilisons ceux qui sont déjà présents dans les clés maîtresses secrète et publique. Nous n'avons alors pas besoin d'en générer de nouveaux. Nous devons seulement rafraîchir les valeurs aléatoires : nous avons uniquement besoin de générer un nouveau vecteur $\mathbf{s}$.

De même que pour la construction précédente, nous allons prouver l'anonymat et la sécurité sémantique.

Montrons la *correctness* de notre construction en commençant par montrer celle de la clé de l'utilisateur $\mathsf{usk}[\mathsf{id}]$ Supposons que $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$ renvoie (K, c) et $\mathsf{USKGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$ renvoie $\mathsf{usk}[\mathsf{id}]$.

Nous avons :

$$e([\mathbf{c}_0]_1, \begin{bmatrix}\mathbf{v}\\ u\end{bmatrix}_2) = \left[(\mathbf{Ar})^\top \cdot \begin{pmatrix}\sum_{i=0}^{\ell} f_i(\mathsf{id})\mathbf{Y}_i\mathbf{t} + \mathbf{y'}\\ \sum_{i=0}^{l} f_i(\mathsf{id})\mathbf{x}_i^\top\mathbf{t} + x'\end{pmatrix}\right]_T$$

$$e([\mathbf{c}_1]_1, [\mathbf{t}]_2) = \left[(\mathbf{Ar})^\top \begin{pmatrix}\sum_{i=0}^{\ell} f_i(\mathsf{id})\mathbf{Y}_i\\ \sum_{i=0}^{l} f_i(\mathsf{id})\mathbf{x}_i^\top\end{pmatrix} \cdot \mathbf{t}\right]_T.$$

En effectuant le quotient des deux couplages précédents, on obtient : $\mathsf{K} = \mathbf{z'} \cdot \mathbf{r}$.
Vérifions maintenant la *correctness* de la clé de traçage $\mathsf{tsk}[\mathsf{id}]$.

Supposons que $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$ renvoie $c = ([\mathbf{c}_0]_1, [\mathbf{c}_1]_1)$ et $\mathsf{TSKGen}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}')$ renvoie $\mathsf{tsk}[\mathsf{id}']$.

Nous obtenons :

$$\frac{e([\mathbf{c}_1]_1, [\mathsf{tsk}_1^\top]_2)}{e([\mathbf{c}_0]_1, [\mathsf{tsk}_2^\top]_2)} \stackrel{?}{=} \frac{\mathbf{r}^\top \sum_{i=0}^{l} f(\mathsf{id})\mathbf{Z}_i^\top \cdot (\mathbf{As})^\top}{(\mathbf{Ar})^\top\mathbf{s}^\top \sum_{i=0}^{l} f(\mathsf{id}')\mathbf{Z}_i^\top} = [\mathbf{r}^\top\mathbf{s}^\top\mathbf{A}^\top \sum_{i=0}^{l}(f(\mathsf{id}) - f(\mathsf{id}'))\mathbf{Z}_i^\top]_T.$$

Après simplification, on peut remarquer que le quotient précédent est égal à 1_T si et seulement si $\mathsf{id} = \mathsf{id}'$.

Preuves de sécurité

Dans cette partie, nous allons prouver que ce protocole AIBET est anonyme et vérifie la sécurité sémantique tout comme la construction de Blazy *et al.* (cf. section 3.1.2). Afin de montrer ces deux propriétés, nous allons effectuer plusieurs jeux de sécurité tout en utilisant la sécurité sémantique des $\mathsf{MAC}_{\mathsf{SPHF}}$.

Commençons par prouver l'anonymat sous l'hypothèse $\mathcal{D}_k - \mathsf{MDDH}$.

Anonymat

Théorème 18. *Sous l'hypothèse* $\mathcal{D}_k - \mathsf{MDDH}$*, notre construction* AIBET *est anonyme. Pour tout adversaire* $\mathcal{A}$*, il existe deux adversaire* $\mathcal{B}_1$*,* $\mathcal{B}_2$ *avec* $\mathcal{T}(\mathcal{B}_1) \approx \mathcal{T}(\mathcal{B}_2) \approx \mathcal{T}(\mathcal{A})$ *et* $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{AIBET}}(\mathcal{A}) \leq \mathsf{Adv}^{prcma}_{[\mathcal{D}_k,\mathsf{GGen}]}(\mathcal{B}_1) + \mathsf{Adv}^{prcma\mathsf{MAC}}_{\mathsf{MAC}_{\mathsf{SPHF}}}(\mathcal{B}_2)$.

Afin de démontrer l'anonymat (ANON-ID-CPA) de notre construction, nous utilisons la sécurité PR-CMA des $\mathsf{MAC}_{\mathsf{SPHF}}$ ainsi que plusieurs lemmes présentés ci-dessous. Comme dans les démonstrations de l'AIBET reposant sur l'IBE de Boyen-Waters, nous utilisons une succession de jeux présentés dans la figure 5.13.

Lemme 6. $\Pr[\mathsf{ANON\text{-}ID\text{-}CPAreal}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1]$.

Démonstration. Montrons qu'il n'y a que des différences de notations entre le jeu $\mathbf{G_0}$ et le jeu $\mathbf{G_1}$.

Considérons $\mathbf{G_0}$ comme le jeux réel *i.e.* aucune modification n'est effectuée. Dans le jeu $\mathbf{G_1}$, nous modifions la simulation de $\mathbf{c}_1^*$ dans l'algorithme $\mathsf{Enc}(\mathsf{id}^*)$ en remplaçant $\mathbf{Z}_i$ et $\mathbf{z}'$ par leurs définitions respectives :

$$\mathbf{c}_1^* = \sum_{i=0}^{\ell} f_i(\mathsf{id}^*)\mathbf{Z}_i\mathbf{r} = \sum_{i=0}^{\ell} f_i(\mathsf{id}^*)(\mathbf{Y}_i^\top \mid \mathbf{x}_i)\mathbf{Ar} = \sum_{i=0}^{\ell} f_i(\mathsf{id}^*)(\mathbf{Y}_i^\top \mid \mathbf{x}_i)\mathbf{c}_0^*.$$

Ainsi, les jeux $\mathbf{G_0}$ et $\mathbf{G_1}$ sont identiques. □

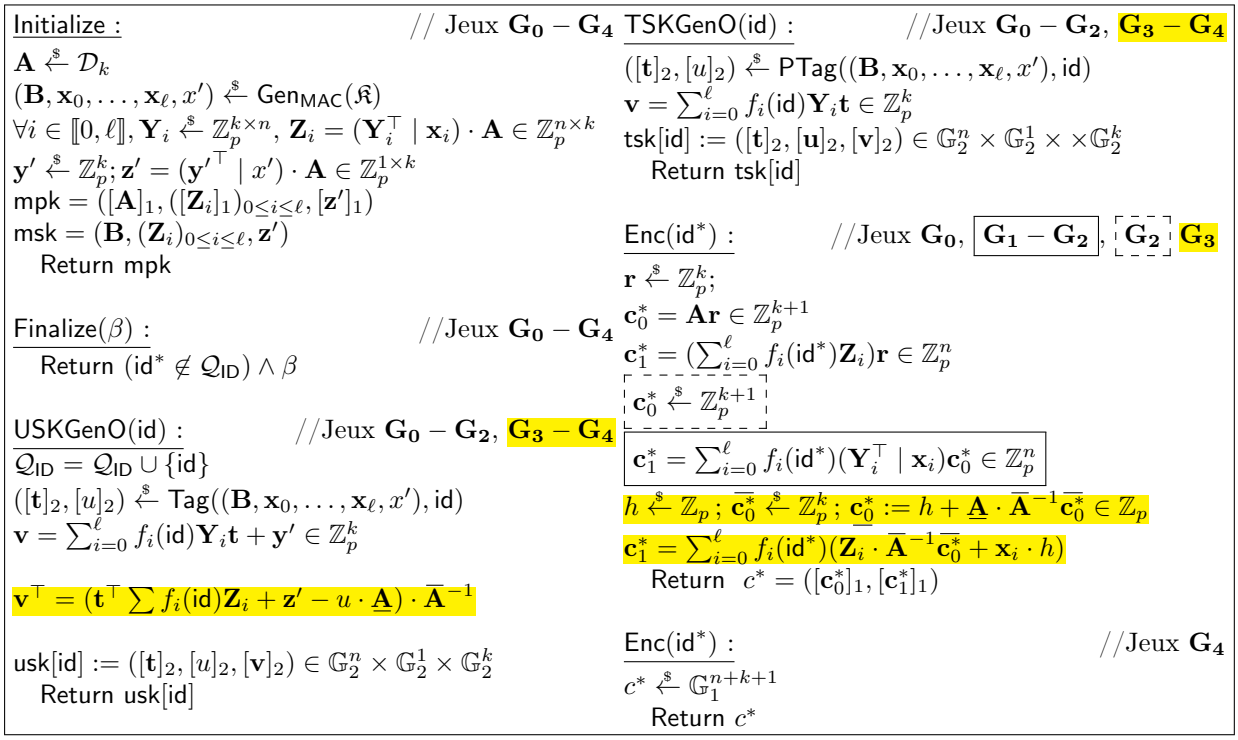


FIGURE 5.13 – Jeux $\mathbf{G_0} - \mathbf{G_4}$ pour la preuve d'anonymat.

Lemme 7. *Il existe un adversaire* $\mathcal{B}_1$ *avec* $\mathsf{TIME}(\mathcal{B}_1) \approx \mathsf{TIME}(\mathcal{A})$ *et* $\mathbf{Adv}_{\mathcal{D}_k, \mathsf{ggen}}(\mathcal{B}_1) \ge |\Pr[\mathbf{G_2}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G_1}^{\mathcal{A}} \Rightarrow 1]|$.

Démonstration. Nous allons montrer que $\mathbf{c}_0^*$ peut être totalement aléatoire.

Nous construisons un adversaire $\mathcal{B}_1$ tel qu'il prenne en entrée $(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{b}]_1)$. Son objectif est de distinguer si $\mathbf{b} = \mathbf{A}\mathbf{w}$ avec $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$ ou si $\mathbf{b}$ est aléatoire.

$\mathcal{B}_1$ simule USKGenO, TSKGenO et Finalize comme dans le jeu $\mathbf{G_1}$. La simulation de Initialize et de Enc est présentée dans la figure 5.14. Il est important de remarquer que $\mathcal{B}_1$ connaît $\mathbf{x}_i, x', \mathbf{Y}_i, \mathbf{y}'$. De plus, $\mathcal{B}_1$ peut générer $([\mathbf{Z}_i]_1)_{0 \le i \le \ell}$ et $[\mathbf{z}']_1$ pour obtenir la clé mpk. Concernant

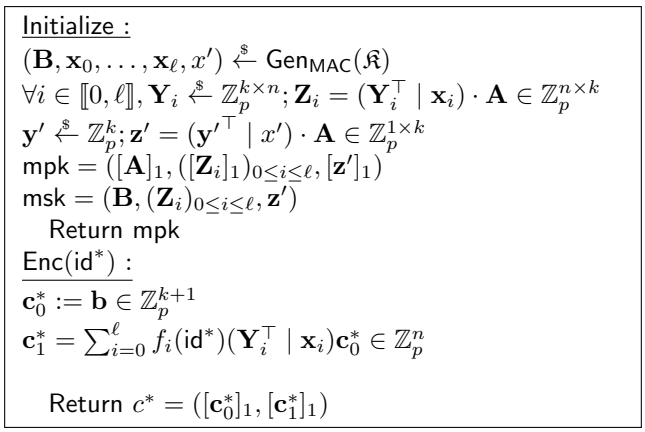


FIGURE 5.14 – Description of $\mathcal{B}_1(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{b}]_1)$ pour la preuve du lemme 7.

les demandes d'encapsulation, il peut également générer $[\mathbf{c}_1^*]_1$ à partir de $[\mathbf{A}]_1$ et de $[\mathbf{b}]_1$.

Si $\mathbf{b} = \mathbf{A}\mathbf{w}$ avec $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$, la simulation est la même que dans le jeu $\mathbf{G_1}$. Si $\mathbf{b}$ est aléatoire alors la simulation est distribuée comme dans le jeu $\mathbf{G_2}$. □

Dans le jeux $\mathbf{G_3}$, on simule les valeurs de $\mathbf{v}$ généré par $\mathsf{USKGenO(id)}$ sans utiliser $(\mathbf{Y}_i)_{0\leq i\leq \ell}$ et $(\mathbf{Y}'_i)_{0\leq i\leq \ell'}$.

Lemme 8. $\Pr[\mathbf{G_3}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathbf{G_2}^{\mathcal{A}} \Rightarrow 1]$.

Démonstration. $\mathbf{G_3}$ n'utilise plus $(\mathbf{Y}_i)_{0\leq i\leq \ell}$ et $\mathbf{Y}'$. En posant $\mathbf{Z}_i = (\mathbf{Y}_i^\top \mid \mathbf{x}_i)\mathbf{A}$, nous avons $\mathbf{Y}_i^\top = (\mathbf{Z}_i - \mathbf{x}_i \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$. De la même manière, on obtient : $\mathbf{Y}'^\top = (\mathbf{z}' - x' \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$.

Il est important de noter que l'on peut calculer la valeur de $[\mathbf{v}]_2$ dans $\mathbb{G}_2$ dès que $\mathbf{A}$, $\mathbf{z}'$ et $\mathbf{Z}_i$ sont explicitement connus et que $[\mathbf{t}]_2$ et $[u]_2$ le sont également.

En ce qui concerne la distribution de $\mathsf{Enc(id^*)}$, on peut voir que $\mathbf{c}_0^*$ est uniformément aléatoire comme dans $\mathbf{G_2}$. En définissant $h = \underline{\mathbf{c}_0^*} - \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*}$, nous avons :

$$\begin{aligned}
\mathbf{c}_1^* &= \sum f_i(\mathsf{id}_i^*)(\mathbf{Z}_i \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*} + \mathbf{x}_i \cdot (\underline{\mathbf{c}_0^*} - \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*})) \\
&= \sum f_i(\mathsf{id}_i^*)(\mathbf{Y}_i^\top \overline{\mathbf{A}} + \mathbf{x}_i\underline{\mathbf{A}}) \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*} + \mathbf{x}_i \cdot (\underline{\mathbf{c}_0^*} - \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*})) \\
&= \sum f_i(\mathsf{id}_i^*)(\mathbf{Y}_i^\top \mid \mathbf{x}_i)\mathbf{c}_0^*
\end{aligned}$$

ainsi $\mathbf{c}_1^*$ est distribué de la même façon que dans $\mathbf{G_2}$. □

Lemme 9. *Il existe un adversaire* $\mathcal{B}_2$ *avec* $\mathsf{TIME}(\mathcal{B}_2) \approx \mathsf{TIME}(\mathcal{A})$, $\mathbf{Adv}^{\mathsf{PRCMA}_{\mathsf{MAC}}}\mathcal{B}_2 \geq |\Pr[\mathbf{G_4}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G_3}^{\mathcal{A}} \Rightarrow 1]|$.

Démonstration. Dans $\mathbf{G_4}$, on répond aux demandes effectuées à $\mathsf{Enc(id^*)}$ en choisissant un c^* aléatoire. On construit un adversaire $\mathcal{B}_2$ (figure. 5.15) afin de montrer que les différences entre les jeux $\mathbf{G_4}$ et $\mathbf{G_3}$ peuvent être majorées par l'avantage de sécurité des $\mathsf{MAC_{SPHF}}$. Intuitivement, la sécurité des $\mathsf{MAC_{SPHF}}$ est la même dans les jeux $\mathbf{G_3}$ et $\mathbf{G_4}$. La clé $\mathsf{sk_{MAC}}$ reste secrète tant que $\mathcal{B}_2$ fait des demandes à $\mathsf{Enc(id^*)}$.

$\mathcal{B}_2$ a accès aux oracles $\mathsf{Initialize_{MAC}}$, Eval, PEval, Chal et $\mathsf{Finalize_{MAC}}$ des jeux définis pour la sécurité $\mathsf{RPR\text{-}CMA}^0$ présentés dans la figure 3.6.

$\underline{\mathsf{Initialize}}$:	$\underline{\mathsf{TSKGenO(id)}}$:
$\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$	$\mathcal{Q}_{\mathsf{ID}} = \mathcal{Q}_{\mathsf{ID}} \cup \{\mathsf{id}\}$
$\forall i \in [\![0,\ell]\!] : \mathbf{Z}_i \xleftarrow{\$} \mathbb{Z}_p^{n\times k}$	$([\mathbf{t}]_2, [u]_2) \xleftarrow{\$} \mathsf{PEval(id)}$
$\mathbf{z}' \xleftarrow{\$} \mathbb{Z}_p^{1\times k}$,	$\mathbf{v}^\top = (\mathbf{t}^\top \sum f_i(\mathsf{id})\mathbf{Z}_i - u \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$
$\mathsf{mpk} := ([\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{0\leq i\leq \ell}, [\mathbf{z}'])$	$\mathsf{tsk[id]} := ([\mathbf{t}]_2, [u]_2, [\mathbf{v}]_2)$
Return mpk	Return tsk[id]
$\underline{\mathsf{USKGenO(id)}}$:	$\underline{\mathsf{Enc(id^*)}}$: //une demande
$\mathcal{Q}_{\mathsf{ID}} = \mathcal{Q}_{\mathsf{ID}} \cup \{\mathsf{id}\}$	$([h]_1, [\mathbf{h}_0]_1, [h_1]_T) \xleftarrow{\$} \mathsf{Chal(id^*)}$
$([\mathbf{t}]_2, [u]_2) \xleftarrow{\$} \mathsf{Eval(id)}$	$\overline{\mathbf{c}_0^*} \xleftarrow{\$} \mathbb{Z}_p^k$, $\underline{\mathbf{c}_0^*} = h + \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*} \in \mathbb{Z}_p$
$\mathbf{v}^\top = (\mathbf{t}^\top \sum f_i(\mathsf{id})\mathbf{Z}_i + \mathbf{z} - u \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$	$\mathbf{c}_1^* = \sum_{i=0}^{\ell} f_i(\mathsf{id^*})\mathbf{Z}_i \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*} + \mathbf{h}_0$
$\mathsf{usk[id]} := ([\mathbf{t}]_2, [u]_2, [\mathbf{v}]_2)$	Return $C^* = ([\mathbf{c}_0^*]_1, [\mathbf{c}_1^*]_1)$
Return usk[id]	$\underline{\mathsf{Finalize}(\beta')}$:
	Return $(\mathsf{id^*} \notin \mathcal{Q}_{\mathsf{ID}}) \wedge \mathsf{Finalize_{MAC}}(\beta')$

FIGURE 5.15 – Description de $\mathcal{B}_2$ pour le Lemme 9.

Si $(\mathbf{h}_0, h_1)$ est distribué uniformément (*i.e.* $\mathcal{B}_2$ est dans le jeu $\mathsf{PRCMArand}$) alors, pour $\mathcal{A}$, le jeu $\mathbf{G_3}$ est identique au jeu $\mathbf{G_4}$. Si $(\mathbf{h}_0, h_1)$ est le vrai couple (i.e., jeu $\mathsf{PRCMAreal}$), alors $\mathcal{A}$ obtient les mêmes informations que dans le jeu $\mathbf{G_3}$. □

Ces différents lemmes nous permettent de conclure la preuve de l'anonymat de notre AIBET. Nous allons maintenant montrer la sécurité sémantique.

Sécurité Sémantique

Théorème 19. *Sous l'hypothèse* $\mathcal{D}_k$ – MDDH, *notre construction* AIBET *sémantiquement sûr. Pour tout adversaire* $\mathcal{A}$, *il existe deux adversaire* $\mathcal{B}_1$, $\mathcal{B}_2$ *avec* $\mathsf{TIME}(\mathcal{B}_1) \approx \mathsf{TIME}(\mathcal{B}_2) \approx \mathsf{TIME}(\mathcal{A})$ *et* $\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{AIBET}}(\mathcal{A}) \leq \mathsf{Adv}^{prcma}_{[\mathcal{D}_k,\mathsf{GGen}]}(\mathcal{B}_1) + \mathsf{Adv}^{prcma\mathsf{MAC}}_{\mathsf{MAC}_{\mathsf{SPHF}}}(\mathcal{B}_2)$.

Sur le même modèle que la preuve de l'anonymat, nous allons utiliser plusieurs lemmes pour démontrer la sécurité sémantique (IND-ID-CPA).

Les différents jeux utilisés dans les preuves suivantes sont présentés dans la figure 5.16.

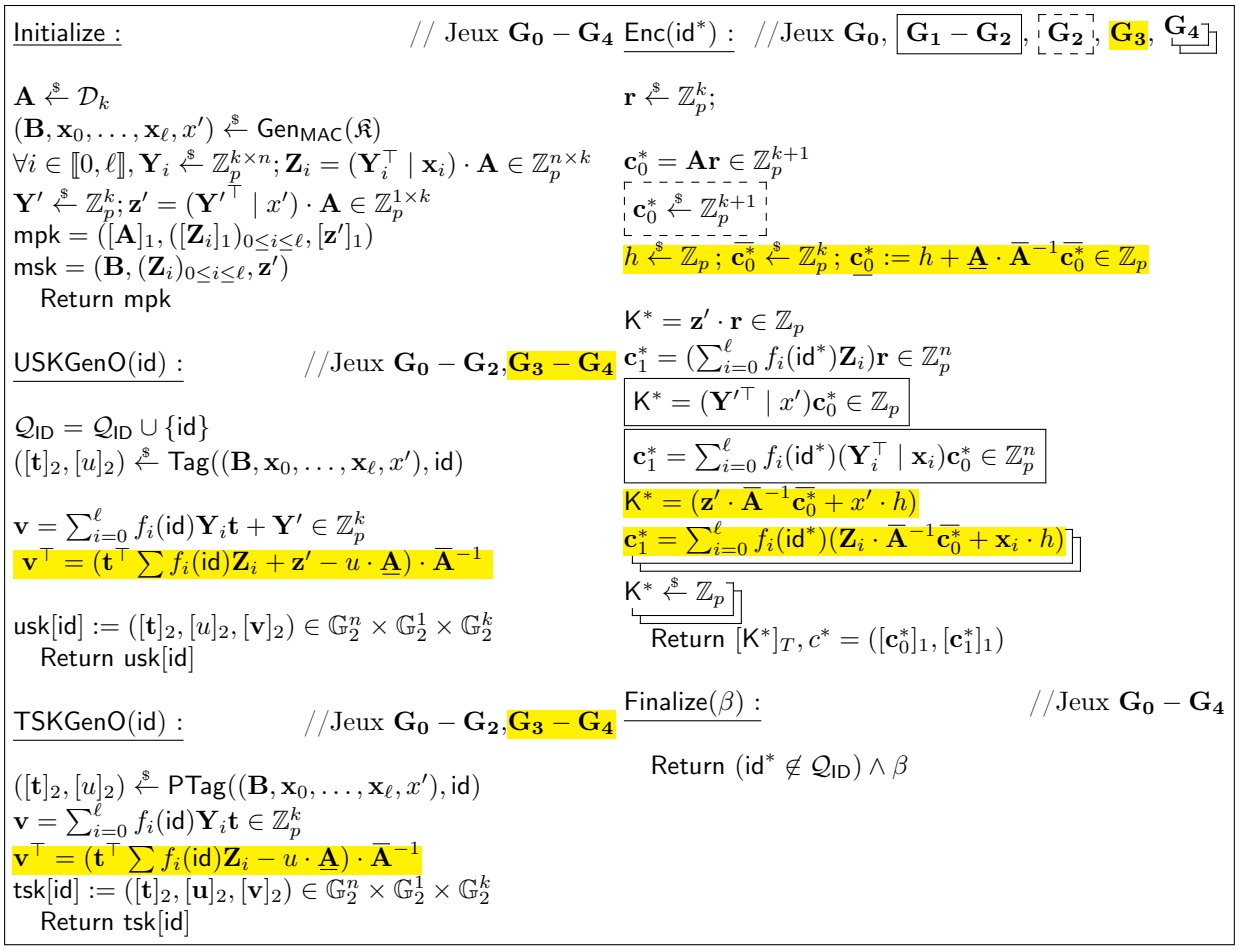


FIGURE 5.16 – Jeux $\mathbf{G_0} - \mathbf{G_4}$ pour la preuve de l'IND-ID-CPA.

Lemme 10. $\Pr[\mathsf{IND\text{-}ID\text{-}CPA}^{\mathcal{A}}_{\mathsf{real}} \Rightarrow 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1]$.

Démonstration. Montrons qu'il n'y a pas de différences entre le jeu $\mathbf{G_0}$ et le jeu $\mathbf{G_1}$.
$\mathbf{G_0}$ est le jeu réel. Dans le jeu $\mathbf{G_1}$, on change la simulation de $\mathbf{c}_1^*$ dans $\mathsf{Enc}(\mathsf{id}^*)$ en substituants $\mathbf{Z}_i$ et $\mathbf{z}'_i$ avec leurs définitions respectives. On obtient :

$$\mathbf{c}_1^* = \sum_{i=0}^{\ell} f_i(\mathsf{id}^*)\mathbf{Z}_i\mathbf{r} = \sum_{i=0}^{\ell} f_i(\mathsf{id}^*)(\mathbf{Y}_i^\top \mid \mathbf{x}_i)\mathbf{A}\mathbf{r} = \sum_{i=0}^{\ell} f_i(\mathsf{id}^*)(\mathbf{Y}_i^\top \mid \mathbf{x}_i)\mathbf{c}_0^*.$$

De même, on simule K^* en utilisant la définition de z' :

$$\mathsf{K}^* = (\mathbf{Y}'^\top | x')\mathbf{A}\mathbf{r} = \mathsf{K}^* = (\mathbf{Y}'^\top | x')\mathbf{c}_0^*.$$

Ces transformations d'écriture n'impactent pas les jeux de sécurité. On peut donc conclure que les jeux $\mathbf{G_0}$ et $\mathbf{G_1}$ sont identiques. □

Lemme 11. *Il existe un adversaire* $\mathcal{B}_1$ *avec* $\mathsf{TIME}(\mathcal{B}_1) \approx \mathsf{TIME}(\mathcal{A})$ *et* $\mathbf{Adv}_{\mathcal{D}_k,\mathsf{ggen}}(\mathcal{B}_1) \geq |\Pr[\mathbf{G_2}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G_1}^{\mathcal{A}} \Rightarrow 1]|$.

Démonstration. Nous allons montrer que la distribution de $\mathbf{G_1}$ correspond au cas réel de $\mathcal{D}_k - \mathsf{MDDH}$ et que la distribution de $\mathbf{G_2}$ correspond au cas aléatoire de $\mathcal{D}_k - \mathsf{MDDH}$. Maintenant, nous construisons un *distinguisher* $\mathcal{B}_1$. Il prend en entrées $(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{b}]_1)$ et il doit distinguer si $\mathbf{b} = \mathbf{Aw}$ avec $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$ ou si $\mathbf{b}$ est uniformément aléatoire. Il simule alors $\mathsf{USKGenO}, \mathsf{TSKGenO}$ et $\mathsf{Finalize}$ de la même manière que dans le jeu $\mathbf{G_1}$.

Initialize :	$\mathsf{Enc}(\mathsf{id}^*)$:
$(\mathbf{B}, \mathbf{x}_0, \ldots, \mathbf{x}_\ell, x') \xleftarrow{\$} \mathsf{Gen}_{\mathsf{MAC}}(\mathfrak{K})$ $\mathcal{G}$ et $[\mathbf{A}]_1$ viennent du challenge $\mathcal{D}_k - \mathsf{MDDH}$. $\forall i \in [\![0, \ell]\!], \mathbf{Y}_i \xleftarrow{\$} \mathbb{Z}_p^{k\times n}; \mathbf{Z}_i = (\mathbf{Y}_i^\top \mid \mathbf{x}_i) \cdot \mathbf{A} \in \mathbb{Z}_p^{n\times k}$ $\mathbf{y}' \xleftarrow{\$} \mathbb{Z}_p^k; \mathbf{z}' = (\mathbf{y}'^\top \mid x') \cdot \mathbf{A} \in \mathbb{Z}_p^{1\times k}$ For $i = 0, \ldots, \ell' : \mathbf{z}'_i \xleftarrow{\$} \mathbb{Z}_p^{1\times k}$ $\mathsf{mpk} = ([\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{0\leq i\leq \ell}, [\mathbf{z}']_1)$ $\mathsf{msk} = (\mathbf{B}, (\mathbf{Z}_i)_{0\leq i\leq \ell}, \mathbf{z}')$ Return mpk	$\mathsf{K}_0^* \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{c}_0^* \xleftarrow{\$} \mathbb{G}_1^{n+k+1}$ $\mathbf{c}_0^* := \mathbf{b} \in \mathbb{Z}_p^{k+1}$ $\mathbf{c}_1^* = \sum_{i=0}^{\ell} f_i(\mathsf{id}^*)(\mathbf{Y}_i^\top \mid \mathbf{x}_i)\mathbf{c}_0^* \in \mathbb{Z}_p^n$ $\mathsf{K}^* = (\mathbf{y}'^\top \mid x')\mathbf{c}_0^* \in \mathbb{Z}_p$ Return $[\mathsf{K}^*]_T, C^* = ([\mathbf{c}_0^*]_1, [\mathbf{c}_1^*]_1)$

FIGURE 5.17 – Description de $\mathcal{B}_1(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{b}]_1)$ pour la preuve du lemme 11.

On peut remarquer que $\mathcal{B}_1$ connaît les données secrètes $\mathbf{x}_i, x', \mathbf{Y}_i, \mathbf{Y}'$. Grâce à cela, il peut calculer $([\mathbf{Z}_i]_1)_{0\leq i\leq \ell}$ et $[\mathbf{z}']_1$ afin de retrouver mpk et de générer $[\mathbf{c}_1^*]_1, [K^*]_T$ pour les demandes d'encapsulation de $[\mathbf{A}]_1$ et $[\mathbf{b}]_1$. Si $\mathbf{b} = \mathbf{Aw}$ pour $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$, alors la simulation est identique au jeu $\mathbf{G_1}$. Si $\mathbf{b}$ est aléatoire, alors la simulation est identique au jeu $\mathbf{G_2}$. □

Pour le jeu $\mathbf{G_3}$, on simule les valeurs de $\mathbf{v}$ calculées dans $\mathsf{USKGenO}(\mathsf{id})$ sans utiliser $(\mathbf{Y}_i)_{0\leq i\leq \ell}$ et $(\mathbf{Y}'_i)_{0\leq i\leq \ell'}$.

Lemme 12. $\Pr[\mathbf{G_3}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathbf{G_2}^{\mathcal{A}} \Rightarrow 1]$.

Démonstration. $\mathbf{G_3}$ ne peut plus utiliser $(\mathbf{Y}_i)_{0\leq i\leq \ell}$ et $\mathbf{Y}'$. En posant, $\mathbf{Z}_i = (\mathbf{Y}_i^\top \mid \mathbf{x}_i)\mathbf{A}$, nous avons $\mathbf{Y}_i^\top = (\mathbf{Z}_i - \mathbf{x}_i \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$, de même : $\mathbf{Y}'^\top = (\mathbf{z}' - x' \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$.

On peut calculer les valeurs de $[\mathbf{v}]_2 \in \mathbb{G}_2$ comme le montre les égalités suivantes..

$$\begin{aligned}
\mathbf{v}^\top &= \left(\mathbf{t}^\top \sum f_i(\mathsf{id})(\mathbf{Z}_i - \mathbf{x}_i \cdot \underline{\mathbf{A}})\right) \overline{\mathbf{A}}^{-1} \\
&= \left(\mathbf{t}^\top \sum f_i(\mathsf{id})\mathbf{Z}_i - (\mathbf{t}^\top \sum f_i(\mathsf{id})\mathbf{x}_i) \cdot \underline{\mathbf{A}}\right) \cdot \overline{\mathbf{A}}^{-1}
\end{aligned}$$

On en déduit que $\mathbf{A}$, $\mathbf{z}'$ et $\mathbf{Z}_i$ sont connus de l'adversaire $\mathcal{A}$.

Comme pour la distribution de $\mathsf{Enc}(\mathsf{id}^*)$, on peut voir que $\mathbf{c}_0^*$ est aléatoire, comme dans le jeu $\mathbf{G_2}$. En définissant $h = \underline{\mathbf{c}_0^*} - \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*}$, nous obtenons :

$$\begin{aligned}
\mathbf{c}_1^* &= \sum f_i(\mathsf{id}_i^*)(\mathbf{Z}_i \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*} + \mathbf{x}_i \cdot (\underline{\mathbf{c}_0^*} - \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*})) \\
&= \sum f_i(\mathsf{id}_i^*)(\mathbf{Y}_i^\top \overline{\mathbf{A}} + \mathbf{x}_i\underline{\mathbf{A}}) \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*} + \mathbf{x}_i \cdot (\underline{\mathbf{c}_0^*} - \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1}\overline{\mathbf{c}_0^*})) \\
&= \sum f_i(\mathsf{id}_i^*)(\mathbf{Y}_i^\top \mid \mathbf{x}_i)\mathbf{c}_0^*
\end{aligned}$$

ainsi $\mathbf{c}_1^*$ est identique au jeu $\mathbf{G_2}$. □

Lemme 13. *Il existe un adversaire* $\mathcal{B}_2$ *avec* $\mathsf{TIME}(\mathcal{B}_2) \approx \mathsf{TIME}(\mathcal{A})$ *et* $\mathbf{Adv}^{\mathsf{PRCMA}_{\mathsf{MAC}}}\mathcal{B}_2 \geq |\Pr[\mathbf{G_4}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G_3}^{\mathcal{A}} \Rightarrow 1]|$.

Démonstration. Dans le jeu $\mathbf{G_4}$, on répond aux demandes de $\mathsf{Enc}(\mathsf{id}^*)$ en choisissant un K^* aléatoire. On construit un adversaire $\mathcal{B}_2$ pour montrer la différence entre les jeux $\mathbf{G_3}$ et $\mathbf{G_4}$. La description de $\mathcal{B}_2$ est présentée dans la figure 5.18. Nous allons montrer qu'en utilisant la sécurité des $\mathsf{MAC}_{\mathsf{SPHF}}$, il possible de majorer l'avantage de $\mathcal{B}_2$ par celui des $\mathsf{MAC}_{\mathsf{SPHF}}$. Nous utilisons la sécurité PR-CMA des $\mathsf{MAC}_{\mathsf{SPHF}}$ comme montré dans la figure 3.5. Ainsi, l'adversaire $\mathcal{A}$ a accès aux algorithmes Initialize, Eval, Chal et Finalize des jeux $\mathsf{PR\text{-}CMA}_{\mathsf{real}}$ et $\mathsf{PR\text{-}CMA}_{\mathsf{rand}}$.

$\mathcal{B}_2$ a accès aux oracles $\mathsf{Initialize}_{\mathsf{MAC}}$, Eval, PEval, Chal et $\mathsf{Finalize}_{\mathsf{MAC}}$ des jeux définis pour la sécurité $\mathsf{RPR\text{-}CMA}^1$ présentés dans la figure 3.7.

Initialize : $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$ $\forall i \in [\![0, \ell]\!] : \mathbf{Z}_i \xleftarrow{\$} \mathbb{Z}_p^{n \times k}$; $\mathbf{z}' \xleftarrow{\$} \mathbb{Z}_p^{1 \times k}$ $\mathsf{mpk} := ([\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{0 \le i \le \ell}, [\mathbf{z}'])$ Return mpk	TSKGen(id) : $\mathcal{Q}_{\mathsf{ID}} = \mathcal{Q}_{\mathsf{ID}} \cup \{\mathsf{id}\}$ $([\mathbf{t}]_2, [u]_2) \xleftarrow{\$} \mathsf{PEval}(\mathsf{id})$ $\mathbf{v}^\top = (\mathbf{t}^\top \sum f_i(\mathsf{id})\mathbf{Z}_i - u \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$ $\mathsf{tsk}[\mathsf{id}] := ([\mathbf{t}]_2, [u]_2, [\mathbf{v}]_2)$ Return tsk[id]

USKGenO(id) :
$\mathcal{Q}_{\mathsf{ID}} = \mathcal{Q}_{\mathsf{ID}} \cup \{\mathsf{id}\}$
$([\mathbf{t}]_2, [u]_2) \xleftarrow{\$} \mathsf{Eval}(\mathsf{id})$
$\mathbf{v}^\top = (\mathbf{t}^\top \sum f_i(\mathsf{id})\mathbf{Z}_i + \mathbf{z} - u \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$
$\mathsf{usk}[\mathsf{id}] := ([\mathbf{t}]_2, [u]_2, [\mathbf{v}]_2)$
Return usk[id]
	Enc(id*) : //une seule requête
	$([h]_1, [\mathbf{h}_0]_1, [h_1]_T) \xleftarrow{\$} \mathsf{Chal}(\mathsf{id}^*)$
	$\overline{\mathbf{c}_0^*} \xleftarrow{\$} \mathbb{Z}_p^k$, $\underline{\mathbf{c}_0^*} = h + \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1} \overline{\mathbf{c}_0^*} \in \mathbb{Z}_p$
	$\mathbf{c}_1^* = \sum_{i=0}^{\ell} f_i(\mathsf{id}^*)\mathbf{Z}_i \cdot \overline{\mathbf{A}}^{-1} \overline{\mathbf{c}_0^*} + \mathbf{h}_0$
	$\mathsf{K}^* = \mathbf{z}' \cdot \overline{\mathbf{A}}^{-1} \overline{\mathbf{c}_0^*} + h_1$
	Return $[\mathsf{K}^*]_T, C^* = ([\mathbf{c}_0^*]_1, [\mathbf{c}_1^*]_1)$
	
	Finalize(β') :
	Return $(\mathsf{id}^* \notin \mathcal{Q}_{\mathsf{ID}}) \wedge \mathsf{Finalize}_{\mathsf{MAC}}(\beta')$

FIGURE 5.18 – Description de $\mathcal{B}_2$ pour le lemme 13.

Si $(\mathbf{h}_0, h_1)$ est uniforme (cas PRCMArand) alors, pour $\mathcal{A}$, le jeu est le même que dans le jeu $\mathbf{G_4}$. Si $(\mathbf{h}_0, h_1)$ est réel cas (PRCMAreal) alors pour $\mathcal{A}$, le jeu est le même que dans le jeu $\mathbf{G_3}$. □

Démonstration. Théorème 11 La démonstration de ce théorème se déduit des différents lemmes précédents tout en remarquant que dans le jeu $\mathbf{G_4}$, les challenge que reçoit $\mathcal{A}$ sont purement aléatoires. □

Ces différents lemmes permettent de conclure la preuve de sécurité pour la sécurité sémantique de notre construction.

Nous avons démontré que notre AIBET conserve les propriétés de sécurité de la construction originale *i.e.* l'anonymat et la sécurité sémantique. Il est intéressant de noter que cette construction est présentée sous les hypothèses $k - \mathsf{MDDH}$. Elle est applicable au cas $k = 1$, ce cas étant le plus utile pour une utilisation concrète de ce protocole.

Dans la suite, nous présentons cette construction *i.e.* l'AIBET sous l'hypothèse SXDH ($k = 1$).

AIBET reposant sur l'hypothèse SXDH

Ici nous présentons la construction précédente pour le cas particulier $k = 1$. Cette version a la particularité de faire intervenir des vecteurs ce qui est un avantage pour une application concrète *i.e.* une implémentation. Dans cette version, l'hypothèse de sécurité utilisée est SXDH. Cette construction, de même que la construction générique, est IND-ID-CPA et ANON-ID-CPA sous l'hypothèse SXDH.

La construction est présentée dans la figure 5.19. Nous rappelons que ce protocole est identique à celui présenté dans la section précédente, les fonctions restent les mêmes que dans l'IBE de Blazy *et al.* tout en conservant nos deux algorithmes TSKGen et TVerify. En ce qui concerne

les preuves de sécurité, il s'agit seulement d'adapter les précédentes, au cas particulier $k = 1$ de l'hypothèse MDDH.

$\underline{\mathsf{Setup}(1^\lambda)}$:

$\mathsf{sk}_{\mathsf{MAC}} = (\mathbf{B}, \mathbf{x}_0, \mathbf{x}_1, x') \xleftarrow{\$} \mathbb{Z}_p^{2\times 3} \times \mathbb{Z}_p$
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{2\times 1}$
$\forall i \in \{0,1\} \mathbf{y}_i \xleftarrow{\$} \mathbb{Z}_p^2; z_i = (\mathbf{y}_i \| \mathbf{x}_i)\mathbf{A} \in \mathbb{Z}_p^2$
$y' \xleftarrow{\$} \mathbb{Z}_p; z' = (y' \| x')\mathbf{A} \in \mathbb{Z}_p$
$\mathsf{mpk} := ([\mathbf{A}]_1, [\mathbf{z}_0]_1, [\mathbf{z}_1]_1, [z']_1)$
$\mathsf{msk} := (\mathsf{sk}_{\mathsf{MAC}}, \mathbf{y}_0, \mathbf{y}_1, y')$
Return $(\mathsf{mpk}, \mathsf{msk})$.

$\underline{\mathsf{USK}_\mathsf{G}(\mathsf{sk}, \mathsf{id})}$:

$s \xleftarrow{\$} \mathbb{Z}_q, \mathbf{t} = \mathbf{B}s$
$u = x' + (\mathbf{x}_0 + \mathsf{id} \cdot \mathbf{x}_1)^\top \mathbf{t}$
$v = y' + (\mathbf{y}_0 + \mathsf{id} \cdot \mathbf{y}_1)^\top \mathbf{t}$
Return $\mathsf{usk}[\mathsf{id}] := ([\mathbf{t}]_2, [u]_2, [v]_2)$

$\underline{\mathsf{TSKGen}(\mathsf{sk}, \mathsf{id})}$:

$s \xleftarrow{\$} \mathbb{Z}_q, \mathbf{t} = \mathbf{B}s$
$u = (\mathbf{x}_0 + \mathsf{id} \cdot \mathbf{x}_1)^\top \mathbf{t}$
$v = (\mathbf{y}_0 + \mathsf{id} \cdot \mathbf{y}_1)^\top \mathbf{t}$
Return $\mathsf{tsk}[\mathsf{id}] := ([\mathbf{t}]_2, [u]_2, [v]_2)$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mathsf{id})}$:

$r \xleftarrow{\$} \mathbb{Z}_p$
$\mathbf{c}_0 := (c_{0,0}, c_{0,1}) = (r, a \cdot r)$
$\mathbf{c}_1 = r(\mathbf{z}_0 + \mathsf{id}\mathbf{z}_1)$
$\mathsf{K} = z' \cdot r.$
Return $\mathsf{K} = [\mathsf{K}]_T$ et $c = ([\mathbf{c}_0]_1, [\mathbf{c}_1]_1) \in \mathbb{G}_1^4$.

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, c)}$:

Vérifie $\mathsf{usk}[\mathsf{id}] = ([\mathbf{t}]_2, [u]_2, [v]_2)$
Vérifie $c = ([\mathbf{c}_0]_1, [\mathbf{c}_1]_1)$
$\mathsf{K} = e([c_{0,0}]_1, [u]_2) \cdot e([c_{0,1}]_1, [v]_2)/(e([c_{1,0}]_1, [t_{1,0}]_2) \cdot e([c_{1,1}]_1, [t_{1,1}]_2))$
Return $\mathsf{K} \in \mathbb{G}_T$.

$\underline{\mathsf{TVerify}(\mathsf{tsk}[\mathsf{id}], \mathsf{id}, c)}$:

Vérifie $\mathsf{tsk}[\mathsf{id}] = ([t]_2, [u]_2, [v]_2)$
Return $1_T = e([c_{0,0}]_1, [u]_2) \cdot e([c_{0,1}]_1, [v]_2)/e([c_1]_1, [t]_2)$

FIGURE 5.19 – AIBET reposant sur SXDH.

5.2.3 Conclusion

Au cours de cette section, nous avons présenté trois constructions d'AIBET :

— la première sur [BW06],
— la seconde sur [BKP14],
— la troisième est une réduction de la seconde au cas particuliers $k = 1$.

Nous les comparons dans le tableau suivant par rapport aux constructions originales.

Schéma	$\lvert c \rvert$	$\lvert \mathsf{tsk} \rvert$	Anonymat	T.	Hypothèse
[BW06]	$5\lvert\mathbb{G}_1\rvert$	$\emptyset$	v	x	DBDH
AIBET	$5\lvert\mathbb{G}_1\rvert$	$3\lvert\mathbb{G}_1\rvert$	v	v	DBDH
[BKP14]	$(n+k+1)\lvert\mathbb{G}_1\rvert$	$\emptyset$	v	x	k-MDDH
AIBET	$(n+k+1)\lvert\mathbb{G}_1\rvert$	$(n+k+1)\lvert\mathbb{G}_2\rvert$	v	v	k-MDDH
AIBET	$3\lvert\mathbb{G}_1\rvert$	$3\lvert\mathbb{G}_2\rvert$	v	v	SXDH

FIGURE 5.20 – Comparaison entre notre construction et les originales.

Ce tableau, nous permet de mettre en évidence que nous conservons les propriétés des constructions originales (*i.e.* anonymat et hypothèse de sécurité) tout en rajoutant notre clé de traçage pour un coût en termes d'élément de groupe restreint.

Dans la suite de ce chapitre, nous allons étudier des signatures électroniques. Nous rappelons (cf. section 2.3.6) que les signatures permettent de garantir l'authenticité d'un message. Cette dernière est une composante essentielle de la sécurité des données.

5.3 Conclusion

Ce chapitre du manuscrit était composé de deux grandes parties : la première se concentrer sur une nouvelle primitive permettant de tracer l'identité du destinataire d'un message dans un IBE. La seconde présentée trois constructions de signatures.

Ce chapitre du manuscrit a présenté un protocole de chiffrement basé sur l'identité qui permet de tracer l'identité du destinataire noté AIBET. L'idée est d'utiliser la partie du chiffré qui ne contient pas le tracer mais seulement l'identité du destinataire. Après avoir présenté la construction générale ainsi que les propriétés de sécurité d'un AIBET (anonymat et sécurité sémantique), deux instanciations furent présentées.

Des deux instanciations présentées, on peut retenir que nous ne modifions pas les paramètres de sécurité de la construction originale *i.e.* l'anonymat et la sécurité sémantique. De même l'ajout de nos deux algorithmes qui permettent le traçage (TSKGen et TVerify), ne nécessitent pas d'éléments de groupes supplémentaires à ceux requis par les constructions. Notre clé de traçage tsk, étant construite sur la clé de l'utilisateur, usk, il nous suffit de rafraîchir les valeurs aléatoires tout en autant de l'usk ce qui est nécessaire pour lever le masque dans le chiffré. Un des points les plus importants de AIBET est que la clé de traçage ne permet pas de déchiffrer le message.

Nous allons maintenant présenter un protocole de signature à trois parties dans le modèle standard.

Chapitre 6

Protocole à trois parties : comment signer en sécurité ?

Cette partie est consacrée à la présentation et à l'étude d'une nouvelle primitive de signature à trois parties : un utilisateur, un token et un serveur. L'idée de séparer des clés de signatures entre deux parties est apparu dans l'article de Boyd [BOY86] de 1986. Après cet article plusieurs propositions de signatures à seuil furent proposées. Dans ces constructions, il n'y a pas de partie plus importante qu'une autre, elles doivent, d'une certaine manière être d'accord sur le message qu'elles signent. Le principe d'une signature à seuil est le partage de la clé de signature entre plusieurs entités.

De manière analogue, il existe des protocoles de chiffrement à seuil. Par exemple, le protocole de chiffrement RSA ou le protocole de ElGamal peuvent être modifié en des protocoles à seuil.

Un point important dans la construction est qu'aucune des trois parties ne peut signer le message seule. Cette signature est appelée par la suite TASS pour *Token Aided Server Signature*.

Le protocole de signature TASS est prouvé sûr dans le modèle standard. Toutefois, il peut être étendu au modèle de la composabilité universelle. Ce modèle de sécurité fut présenté par Canetti dans [Can01]. Dans le cas de la sécurité UC, un protocole doit se comporter de manière idéale dans le monde idéal et dans le monde réel.

Un protocole de chiffrement entre trois parties est en cours de soumission à la conférence PKC 2020.

Sommaire

6.1 Principe de la signature

Les signatures à trois parties ou plus sont des primitives déjà existantes de la cryptographie. On peut notamment citer le protocole de signature Pass2Sign de Camenisch, Lehmann, Neven et Samelin [CLNS16]. Ce protocole est prouvé sûr dans le modèle de la composabilité universelle.

Dans cette signature, nous considérons trois parties : l'utilisateur $\mathcal{U}$, un token $\mathcal{T}$ et un serveur $\mathcal{S}$. L'utilisateur fournit le message que vont signer de manière collaborative le token et le serveur.

L'utilisateur possède un mot de passe $\mathsf{pw}_{\mathcal{U}}$ qui permet au serveur de vérifier que la requête émane bien de l'utilisateur et non d'une tierce personne. Comme dans une signature en blanc, le serveur ne doit pas apprendre d'information sur le message qu'il signe. De même, le token et le serveur ne doivent pas pouvoir générer une signature valide l'un sans l'autre.

Les smartphones ou tablettes sont utilisés quotidiennement. De part ces utilisations, ils sont amenés à traiter une grande quantité de données sensibles ou non. Afin de faciliter l'utilisation de services sécurisés ou l'envoie de données sensibles, un protocole de signature utilisant un token, dont on n'effectue aucune hypothèse sur la sécurité peut être envisagé. Ce protocole permet, dans le modèle standard, d'effectuer une signature valide entre l'utilisateur, le token et le serveur. On peut noter que le serveur peut être accessible depuis un ordinateur public sans sécurité supplémentaire.

Avant le commencement du protocole de signature, l'utilisateur et le serveur échange des mots de passe afin de pouvoir s'authentifier l'un à l'autre. Pour effectuer cet échange, ils utilisent un PAKE (*Password Authentification Key Exchange*). La définition d'un PAKE ainsi qu'une précision sur son utilisation sont présentées dans la suite.

6.2 Définitions

Avant de présenter notre schéma, voici une définition des cinq protocoles utilisés. Nous notons par $\mathsf{ISign}\langle\mathcal{U},\mathcal{S},\mathcal{T}\rangle$ le protocole interactif de signature entre l'utilisateur $\mathcal{U}$, le serveur $\mathcal{S}$ et le token $\mathcal{T}$.

Définition 38. *Un schéma de signature* Token-Server-Aided *est composé de quatre algorithmes* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ISign}\langle\mathcal{U},\mathcal{S},\mathcal{T}\rangle, \mathsf{Verif})$ *:*

- $\mathsf{Setup}(\mathfrak{K})$: *génère les paramètres globaux du système* param.
- $\mathsf{KeyGen}(\mathsf{param})$ *: génère une paire de clé* $(\mathsf{vk},\mathsf{sk})$ *de signature et de vérification.*
- $\mathsf{KeyShare}(\mathsf{sk})$: *génère les clés utilisées par le token* $\mathcal{T}$ *et le serveur* $\mathcal{S}$ $(\mathsf{sk}_{\mathcal{T}},\mathsf{sk}_{\mathcal{S}})$.
- $\mathsf{ISign}\langle\mathcal{U}(\mathsf{vk},pw,m),\mathcal{S}(\mathsf{sk}_{\mathcal{S}},pw),\mathcal{T}(\mathsf{sk}_{\mathcal{T}})\rangle$: *un protocole interactif qui exécute les algorithmes* $\mathcal{S}$*,* $\mathcal{T}$ *et* $\mathcal{U}$*, et qui génère une signature* σ *pour* $\mathcal{U}$.
- $\mathsf{Verif}(\mathsf{vk},m,\sigma)\rightarrow\{0,1\}$ *renvoie 1 si la signature est valide, 0 sinon.*

L'algorithme KeyShare permet de générer une clé de signature pour le token et une clé de signature pour le serveur.

L'algorithme ISign est le protocole interactif entre les trois parties qui permet de générer une signature sur un message m fournit par l'utilisateur.

6.3 Modèle de sécurité

Notre nouvelle primitive doit vérifier des notions de sécurité qui permettent de protéger l'utilisateur contre un système malveillant mais également le système contre un utilisateur malicieux. Pour cela, elle doit vérifier quatre propriétés :

- *Correctness* : Pour tout $\mathfrak{K}\in\mathbb{N}$, $\mathsf{param}\rightarrow\mathsf{Setup}(1^{\mathfrak{K}})$, pour tout couple $(\mathsf{vk},\mathsf{sk})\leftarrow\mathsf{KeyGen}(\mathsf{crs})$, tout couple $(\mathsf{sk}_{\mathcal{S}},\mathsf{sk}_{\mathcal{T}})\leftarrow\mathsf{KeyShare}(\mathsf{sk})$, pour tout message $m\in\mathcal{M}$ et mot de passe $pw\in\mathcal{P}$, notre TASS est correcte si :

$$(\sigma,\perp,\perp)\leftarrow\mathsf{ISign}\langle\mathcal{U}(\mathsf{vk},pw,m),\mathcal{S}(\mathsf{sk}_{\mathcal{S}},pw),\mathcal{T}(\mathsf{sk}_{\mathcal{T}})\rangle,\ \text{où}\ \ \mathsf{Verif}(\mathsf{crs},\mathsf{vk},m,\sigma)=1.$$

- Résistance aux contrefaçons : un adversaire ne doit pas pouvoir forger une n-ième signature valide après en avoir reçu $n-1$ comme on peut le voir dans la figure suivante :
- *Weak Blindness[Fis06]* : un serveur malintentionné $\mathcal{S}^*$ ne doit pas pouvoir déterminer quels messages m_0 ou m_1 fut signé en premier. Le jeu de sécurité est présenté dans la figure suivante :

$\mathsf{Exp}^{\mathsf{Forge}}_{\mathsf{TASS},\mathcal{U}^*}(\mathfrak{K})$:
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $\mathsf{vk}, \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{crs})$
3. $\mathsf{sk}_{\mathcal{S}}, \mathsf{sk}_{\mathcal{T}} \leftarrow \mathsf{KeyShare}(\mathsf{sk})$
4. $(m_1, \sigma_1), \ldots, (m_n, \sigma_n) \leftarrow \mathcal{U}^{*\langle \cdot, \mathcal{S}(\mathsf{sk}_{\mathcal{S}} pw_{\mathcal{S}}[\mathcal{U}]), \mathcal{T}(\mathsf{sk}_{\mathcal{T}})\rangle}(\mathsf{vk}, pw_{\mathcal{U}})$
5. Return 1 ssi
$\forall\, 1 \leq i \leq j \leq n,\ m_i \neq m_j$, et $\mathsf{Verif}(\mathsf{vk}, m_i, \sigma_i) = 1$
et au plus $n-1$ interactions avec $\langle \cdot, \mathcal{S}(\mathsf{sk}_{\mathcal{S}}), \mathcal{T}(\mathsf{sk}_{\mathcal{T}})\rangle$ sont effectuées.

FIGURE 6.1 – Résistance aux contrefaçons.

$\mathsf{Exp}^{\mathsf{Blind}}_{\mathsf{TASS},\mathcal{S}^*}(\mathfrak{K})$:
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $\mathsf{vk}, \mathsf{sk}_{\mathcal{S}}, \mathsf{sk}_{\mathcal{T}}, m_0, m_1 \leftarrow \mathcal{S}^*(\mathsf{param})$
3. $b \leftarrow \{0, 1\}$
4. $(\sigma_b, \perp, \perp) \leftarrow \langle \mathcal{U}(\mathsf{vk}, pw_{\mathcal{U}}, m_b), \mathcal{S}^*(\mathsf{sk}_{\mathcal{S}}, pw_{\mathcal{S}}[\mathcal{U}]), \mathcal{T}(\mathsf{sk}_{\mathcal{T}})\rangle$
5. $(\sigma_{1-b}, \perp, \perp) \leftarrow \langle \mathcal{U}(\mathsf{vk}, pw_{\mathcal{U}}, m_{1-b}), \mathcal{S}^*(\mathsf{sk}_{\mathcal{S}}, pw_{\mathcal{S}}[\mathcal{U}]), \mathcal{T}(\mathsf{sk}_{\mathcal{T}})\rangle$
6. $b^* \leftarrow \mathcal{S}^*(\sigma_0, \sigma_1)$
7. Si $\sigma_0 = \perp$ ou $\sigma_1 = \perp$ alors renvoie un bit aléatoire.
8. Si $b = b^*$ Return 1 sinon Return 0.

FIGURE 6.2 – *Weak Blindness.*

— Signature à seuil : cette propriété permet d'éviter qu'un serveur ou un token puisse générer une signature sans l'autre partie.

$\mathsf{Exp}^{\mathsf{ShareSign}}_{\mathsf{TASS},\mathcal{S}^*,\mathcal{T}^*}(\mathfrak{K})$:
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $\mathsf{vk}, \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{crs})$
3. $\mathsf{sk}_{\mathcal{S}}, \mathsf{sk}_{\mathcal{T}} \leftarrow \mathsf{KeyShare}(\mathsf{sk})$
4. Pour $1 \leq i \leq k$:
$((m_i, \sigma_i), \perp, \perp) \leftarrow \langle \mathcal{U}(\mathsf{vk}, pw), \mathcal{S}^*(\mathsf{sk}_{\mathcal{S}}, pw), \mathcal{T}(\mathsf{sk}_{\mathcal{T}})\rangle$
$((m'_i, \sigma'_i), \perp, \perp) \leftarrow \langle \mathcal{U}(\mathsf{vk}, pw), \mathcal{S}(\mathsf{sk}_{\mathcal{S}}, pw), \mathcal{T}^*(\mathsf{sk}_{\mathcal{T}})\rangle$
5. $((m^*, \sigma^*), \perp) \leftarrow \langle \mathcal{U}(\mathsf{vk}, pw), \mathcal{S}^*(\mathsf{sk}_{\mathcal{S}}, pw)\rangle$
6. $((m^{\times}, \sigma^{\times}), \perp) \leftarrow \langle \mathcal{U}(\mathsf{vk}, pw, m_b), \mathcal{T}^*(\mathsf{sk}_{\mathcal{T}})\rangle$
7. Return 1 si
$\forall\, 1 \leq i \leq k,\ m_i \neq m^*$, et $\mathsf{Verif}(\mathsf{vk}, m^*, \sigma^*) = 1$
ou si $\forall\, 1 \leq i \leq k,\ m'_i \neq m^{\times}$, et $\mathsf{Verif}(\mathsf{vk}, m^{\times}, \sigma^{\times}) = 1$.
sinon Return 0.

FIGURE 6.3 – *Signature à seuil.*

6.4 Construction Générale

Avant de présenter notre construction générale, introduisons les différents schémas cryptographiques que nous utilisons.

6.4.1 Protocoles utilisés dans la construction

Chiffrement de Cramer-Shoup étiqueté

Dans notre protocole, nous utilisons une variante du chiffrement de Cramer-Shoup [CS98] présenté dans la section 3.1.1. Au protocole original de Cramer-Shoup est ajouté un label, cette version est connue sous le nom de *Labeled Short Cramer-Shoup encryption* proposée par Abdalla, Benhamouda et Pointcheval dans [ABP15]. La définition des protocoles de chiffrement labellisés est rappelée dans la section 2.2.

Dans notre construction, ce chiffrement est noté sous la forme $\mathsf{CS.Encrypt}$.

On note par $\mathcal{H}$ l'ensemble des fonctions de hachage résistantes aux collisions.

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}) :}$
$\mathsf{param} \xleftarrow{\$} \mathcal{G}$
Return param

$\underline{\mathsf{KeyGen}(\mathsf{param}) :}$
$\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$
Posons : $c = [x_1 + x_2], d = [y_1 + y_2]$ et $h = [z]$
$H_K \leftarrow \mathcal{H}$
$\mathsf{ek} = (c, d, H_K)$
Return dk, ek

$\underline{\mathsf{Enc}(\ell, \mathsf{ek}, m; r) :}$
$r \xleftarrow{\$} \mathbb{Z}_p$
$\xi = H_K(\ell, r, e)$
$C = (\ell, \mathbf{u}, e = [m + hr], v = [(c + d\xi)r])$
Return C

$\underline{\mathsf{Dec}(\ell, \mathsf{dk}, C) :}$
Calcule $\xi = H_K(\ell, r, e)$ et vérifie quand :
$rx_1 + r\xi y_1 + rx_2 + r\xi y_2 \stackrel{?}{=} v$
Calcule $m = [e - rz]$ ou renvoie $\perp$

FIGURE 6.4 – Chiffrement de Cramer-Shoup étiqueté.

Concernant la sécurité, ce protocole est $\mathsf{IND\text{-}PCA}$ sous les hypothèses DDH et de résistance aux collisions de la fonction de hachage utilisée.

Signature de Waters Asymétrique

Dans le chapitre des définitions 3.2.1, la signature de Waters est présentée. Pour la construction TASS, nous avons besoin d'utiliser la signature de Waters asymétrique. Au lieu de travailler avec un seul groupe $\mathbb{G}$, nous travaillons avec deux groupes $\mathbb{G}_1$ et $\mathbb{G}_2$.

On rappelle la définition de la fonction de Waters.

Définition 39. *On suppose l'existence d'une liste de générateurs indépendants* $[\mathbf{u}_i] \xleftarrow{\$} \mathbb{G}$. *On définit la fonction de Waters comme suit :* $[\mathcal{F}(m)] = [\mathbf{u}_0 \sum_{i=1}^{k} \mathbf{u}_i^{m_i}]$ *avec* $m \in \{0,1\}^k$.

Définition 40. *La signature de Waters asymétrique est composée des quatre algorithmes suivants :*

- $\mathsf{Setup}(\mathfrak{K})$ *: génère les paramètres globaux du système* param.
- $\mathsf{KeyGen}(\mathsf{param})$ *: avec* $x \xleftarrow{\$} \mathbb{Z}_p$, *calcule* $\mathsf{pk} = ([x]_1, [x]_2)$ *et* $\mathsf{sk} = [hx]$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: *prend* $s \xleftarrow{\$} \mathbb{Z}_p$ *et calcule* $\sigma = (\sigma_1, \sigma_2, \sigma_3) = ([\mathsf{sk}s\mathcal{F}(m)], [-s]_1, [-s]_2)$.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: *vérifie quand* $[\sigma_1 + \mathcal{F}(m)\sigma_3]_T \stackrel{?}{=} [x]_T$ *et* $[\sigma_2]_T \stackrel{?}{=} [h\sigma_3]_T$.

6.4.2 Protocole Générique

Maintenant que le modèle de sécurité est posé, nous pouvons présenter d'abord une construction générique puis une concrète qui utilise notamment la signature de Waters asymétrique. Des SPHF sont également utilisées au cours de notre protocole, elles permettent d'assurer la validité de la signature.

Nous notons par Sign le protocole de signature utilisé dans notre construction et par E le protocole de chiffrement.

— $\mathsf{Setup}(1^\lambda)$: Fais appel à $\mathsf{param}_{\mathsf{Sign}} \leftarrow \mathsf{Sign.Setup}(1^\mathfrak{K})$ et $\mathsf{param}_{\mathsf{E}} \leftarrow \mathsf{E.Setup}(1^\mathfrak{K})$. Si les paramètres sont compatibles, alors $\mathsf{param} := (\mathsf{param}_{\mathsf{Sign}}, \mathsf{param}_{\mathsf{E}})$ sont les paramètres publics du système. Ensuite, il génère $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{E.KeyGen}(\mathsf{param}_{\mathsf{E}})$ et chiffre ek dans le crs. La clé de déchiffrement dk n'est pas utilisée.

— $\mathsf{KeyGen}(\mathsf{param})$: Effectue deux appels à $\mathsf{BS.KeyGen}(\mathsf{param}_{\mathsf{Sign}})$ pour obtenir $(\mathsf{vk}_1, \mathsf{sk}_1)$ et $(\mathsf{vk}_2, \mathsf{sk}_2)$. Grâce à la propriété de linéarité de BS, posons $\mathsf{sk} := \mathsf{sk}_1 + \mathsf{sk}_2$ et $\mathsf{vk} := \mathsf{vk}_1 + \mathsf{vk}_2$. Chiffre vk dans le CRS, transmet sk_1 à $\mathcal{T}$ et sk_2 à $\mathcal{S}$.

— $\mathsf{ISign}\langle \mathcal{U}(M, \mathsf{pw}; \rho), \mathcal{T}(\mathsf{sk}_1; \tau), \mathcal{S}(\mathsf{sk}_2, \mathsf{pw}; \xi)\rangle$: protocole interactif entre l'utilisateur $\mathcal{U}$, le token $\mathcal{T}$ et le serveur $\mathcal{S}$, chaque partie utilise respectivement les valeurs aléatoires ρ, τ et ξ.

  — $\mathcal{U}$ calcule $c_{\mathsf{pw}} = \mathsf{E.Enc}(\mathsf{pw}; \rho)$ et envoie (c_{pw}, M) à $\mathcal{T}$.

  — $\mathcal{T}$ envoie c_{pw} à $\mathcal{S}$.

  — $\mathcal{S}$ calcule une SPHF, de la clé de hachage $\mathsf{hk}_{\mathsf{pw}}$ et de celle de projection $\mathsf{hp}_{\mathsf{pw}}$.Dès que $\mathcal{S}$ connaît pw et $\mathsf{hk}_{\mathsf{pw}}$, il peut calculer $H := \mathsf{Hash}(\mathsf{hk}_{\mathsf{pw}}, \mathfrak{L}(\mathsf{ek}, M), c_{\mathsf{pw}})$.

  — $\mathcal{T}$ et $\mathcal{S}$ utilise le protocole interactif $\mathsf{Sign}\langle \mathcal{T}(M), \mathcal{S}(\mathsf{sk}_2)\rangle$, où $\mathcal{S}$ masque le dernier *flow* envoyer par $\mathcal{T}$ avec H.

  — $\mathcal{T}$ utilise la linéarité de BS pour modifier le dernier *flow* reçu et $\mathsf{hp}_{\mathsf{pw}}$ avec sa clé de signature sk_1 (ainsi, le message est signé sous la clé de signature $\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2$). Alors, il randomise le tout en utilisant τ. On note la clé de projection associée $\mathsf{hp}'_{\mathsf{pw}}$ et le *flow* F'. $\mathcal{T}$ envoie F' et $\mathsf{hp}'_{\mathsf{pw}}$ à $\mathcal{U}$.

  — Dès que $\mathcal{U}$ connaît l'aléatoire ρ, il peut générer c_{pw}, il calcule
  $H' := \mathsf{ProjHash}(\mathsf{hp}'_{\mathsf{pw}}, \mathfrak{L}(\mathsf{ek}, M), c_{\mathsf{pw}}, \rho)$. Il l'utilise pour obtenir σ à partir de F'. Si toutes les parties suivent le protocole de signature, σ est valide sur m avec la clé $\mathsf{vk}_1 + \mathsf{vk}_2$.

— $\mathsf{Verif}(\sigma, m, \mathsf{vk})$: renvoie le résultat de $\mathsf{Sign.Verify}(\sigma, m, \mathsf{vk})$.

6.5 Protocole Concret

Dans la figure 6.5, nous présentons une instanciation qui repose sur la construction générale présentée précédemment. Ce protocole utilise la signature de Waters en asymétrique dont la définition est présentée ci-dessus (définition 40).

6.6 Preuves de sécurité

Commençons par montrer que notre signature est correcte.

En utilisant les notations utilisées dans la vérification, nous supposons que l'exécution est honnête, donc que $H = H'$. Nous notons $\oplus$ comme $\cdot$ dans l'objectif de simplifier nos notations. Rappelons les notations utilisées dans notre construction :

$$r = \sum_i r_i, \quad \mathsf{sk} = \mathsf{sk}_1 \cdot \mathsf{sk}_2 = [hx]_1, \quad \mathsf{vk} = [x]_2 \text{ et } \mathcal{F}(M) = [u_0 \sum_i m_i \cdot u_i]_1.$$

$\underline{\mathsf{Setup}(1^{\mathfrak{K}}) :}$

Génère : $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$,
et les générateurs $([u_i]_1)_{[\![0,\ell]\!]}$
$\mathcal{F}(m) = [u_0 \sum_i m_i \cdot u_i]_1$
où $m = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$.
Choisi $[h]_1$
Pose $\mathsf{ek} = [f]_1$.
Return $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, \mathsf{ek}, [h], [\mathbf{u}])$.

$\underline{\mathsf{KeyGen}(\mathsf{param}) :}$

$x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$, set $x = x_1 + x_2$
$(\mathsf{sk}_1, \mathsf{sk}_2) = ([x_1 \cdot h]_1, [x_2 \cdot h]_1)$
$(\mathsf{vk}_1, \mathsf{vk}_2) = ([x_1]_2, [x_2]_2)$
On a $\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2 = [x_1 + x_2 \cdot h]_1 = [x \cdot h]_1$
et $\mathsf{vk} = \mathsf{vk}_1 + \mathsf{vk}_2 = [x]_2$.

$\underline{\mathsf{Verify}(\mathsf{vk}, m, \Sigma) :}$
Si $[\Sigma_1]_T = [h\mathsf{vk} + \mathcal{F}(m)\Sigma_{2,2}]_T$
et $[\Sigma_{2,1}]_T = [\Sigma_{2,2}]_T$
Return 1
Sinon Return 0.

$\underline{\mathsf{ISign}(\langle \mathcal{U}, m, pw_{\mathcal{U}} \rangle, \langle \mathcal{T}, \mathsf{sk}_1 \rangle, \langle \mathcal{S}, \mathsf{sk}_2, pw_{\mathcal{S}}[\mathcal{U}] \rangle)}$
$\mathcal{U}$ commence le protocole :
Pour $i \in [\![1, \ell]\!]$:
$r_i \xleftarrow{\$} \mathbb{Z}_p$
$b_i \leftarrow \mathsf{Enc}(\mathsf{ek}, [m_i \cdot u_i]_1; r_i) = ([r_i, r_i \cdot f + m_i \cdot u_i]_1)$
$r := \sum_i r_i$, $c_{\mathsf{pw}} \leftarrow \mathsf{CS.Encrypt}(\mathsf{pw}, r_{\mathsf{pw}})$
$\mathcal{U}$ envoie $(\{b_i\}_{i \in [\ell]}, c_{\mathsf{pw}})$ à $\mathcal{S}$.

Ensuite $\mathcal{S}$:
Génère hk, hp pour le langage de chaque b_i et chiffre le mot de passe valide c_{pw}.
Calcule $H \leftarrow \mathsf{Hash}(\mathsf{hk}_{\mathsf{pw}}, c_{\mathsf{pw}}) \sum \mathsf{Hash}(\mathsf{hk}_i, b_i)$
$s \xleftarrow{\$} \mathbb{Z}_p$; Calcule :
$\sigma = (\sigma_1, \sigma_2, \sigma_3) = [H + (\mathsf{sk}_1 + s(u_0 \sum b_{i,2}))]_1, ([s]_1, [s]_2), [s \cdot f]_1$.
$\mathcal{S}$ envoie (hp, σ) à $\mathcal{T}$.

$\mathcal{T}$ calcule $\sigma' = (\sigma'_1, \sigma'_2, \sigma'_3) = (\sigma_1 + \mathsf{sk}_2, \sigma_2, \sigma_3)$.
$\mathcal{T}$ envoie (hp, σ') à $\mathcal{U}$

$\mathcal{U}$ récupère $\Sigma_1 = (\sigma'_1 \ominus H')/\sigma'^{r}_3$, et $\Sigma_2 = \sigma'_2$ avec $\mathsf{hp}, r,$ and r_{pw}.

FIGURE 6.5 – Protocole concret de notre TASS.

Premièrement, nous avons l'égalité suivante :

$$\begin{aligned}
[\Sigma_1, g_2]_T &= [\sigma'_1 + H - \sigma'^{r}_3]_T \\
&= [\sigma_1 + \mathsf{sk}_2 + H - \sigma^r_3]_T \\
&= [\mathsf{sk}_1 + \mathsf{sk}_2 + s(u_0 \textstyle\sum b_{i_2}) - fsr]_2 \\
&= [\mathsf{sk}_1 + \mathsf{sk}_2 + s(u_0 \textstyle\sum fr_i + u_i m_i) - fsr]_T \\
&= [\mathsf{sk} + s(u_0 \textstyle\sum fr_i + u_i M_i) - fsr]_2 \\
&= [\mathsf{sk} + s\mathcal{F}(m)] \\
&= [hx + s\mathcal{F}(m)]
\end{aligned}$$

Ensuite nous avons :

$$[h\mathsf{vk} + \mathcal{F}(m)\Sigma_{2,2}]_T = [hh_2 x + s\mathcal{F}(m)]_T$$

On remarque que les termes sont égaux, on peut donc en conclure que notre signature est correcte.

Théorème 20. *Notre protocole* TASS *est résistante aux contrefaçons sous l'hypothèse* CDH+.

Le schéma suivant (figure 6.6) est une vue d'ensemble des différentes modifications que l'on effectue dans la preuve. L'objectif de la démonstration est de montrer que le challenger $\mathcal{B}$ ne peut pas forger de nouvelles signatures sans générer une nouvelle signature de Waters.

Démonstration. Cette preuve est une adaptation de la preuve de [BFPV11]. Dans cet article, les auteurs prouvent la résistance aux contrefaçons de la version asymétrique de la signature de Waters. On construit un simulateur $\mathcal{B}$ qui va jouer les jeux de sécurité associé à CDH+.

$\underline{\mathsf{Setup}(\mathfrak{K})}$:

$\mathsf{param} \leftarrow \mathsf{Setup}(\mathfrak{K})$
$x, y \xleftarrow{\$} \mathbb{Z}_p$, $h_1 = [y]_1$
Return $(\mathsf{param}, [yx])$

$\underline{\mathsf{Sign}(h_1, m)}$:

$H = -2jq_s + y_0 + \sum_i y_i M_i$
$J = z_0 + \sum_i z_i M_i$
$\mathcal{F}(m) = [yH + J]_1$
$\sigma = ([-xJ/H + s\mathcal{F}(m)]_1, [-s + x/H]_1, [-s + x/H]_2)$
Return σ

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, \sigma_{\mathsf{SRC}})}$

Si $H \equiv 0 \mod p$, on arrête.
Sinon : $t = s - x/H$ et
$\sigma = ([yx + t\mathcal{F}(m)]_1, [-t]_1, [-t]_2)$

FIGURE 6.6 – Schéma simplifié de la preuve de résistance aux contrefaçons.

— Tout d'abord, $\mathcal{B}$ reçoit un challenge CDH+ : $([x]_1, [x]_2, [y]_1)$. Le but est de trouver $[xy]_1$.
— $\mathcal{B}$ utilise l'algorithme Setup. h, $\mathbf{u} = (u_i)_{i \in \{0,\dots,l\}} \in \mathbb{G}^{l+1}$ pour la fonction de Waters sont rendus publiques. On pose également $h_1 = [y]_1$ qui joue le rôle de la clé de signature dans ce jeu.
— $\mathcal{B}$ fait tourner KeyGen : il publie $\mathsf{vk} = [x]_1$. Retrouver la clé de signature $\mathsf{sk} = [xy]_1$ est le but de ce jeu.
— $\mathcal{A}$ peut effectuer $k - 1$ requêtes à $\mathcal{B}$ en utilisant le protocole interactif de signature Sign :
  — $\mathcal{B}$ calcule $H = -2jq_s + y_0 + \sum_i y_i m_i$, $J = z_0 + \sum_i z_i m_i$ et $\mathcal{F}(m) = [yH + J]_1$.
  Si $H \equiv 0 \mod p$, on arrête, sinon $\sigma = ([-xJ/H + s\mathcal{F}(m)]_1, [x/H - s]_1, [x/H - s]_2)$.
  σ est une signature valide en posant $t = s - x/H$, on obtient :

$$\begin{aligned}\sigma &= ([-xJ/H + s\mathcal{F}(m)]_1, [x/H - s]_1, [x/H - s]_2)\\ &= ([-xJ/H + syH + sJ]_1, [x/H - s]_1, [x/H - s]_2)\\ &= ([-xJ/H + yx + xJ/H + t\mathcal{F}(m)]_1, [-t]_1, [-t]_2)\\ &= ([yx + t\mathcal{F}(m)]_1, [-t]_1, [-t]_2).\end{aligned}$$

  — $\mathcal{B}$ envoie honnêtement cette signature en utilisant la SPHF.
— Après k requêtes, $\mathcal{A}$ génère une signature σ^* sur un nouveau message m^*.
— $\mathcal{B}$ calcule H et J pour m^*.
— Si $H \not\equiv 0 \mod p$, $\mathcal{B}$ abandonne. Sinon, $\sigma^* = ([xy(\mathcal{F}(m^*)t^*)]_1, [-t^*]_1, [-t^*]_2)$. En calculant $[\sigma_1^* + (\sigma_2^*) + J^*]_1 = [xy]_1$, $\mathcal{B}$ résout le challenge de CDH+.

□

Théorème 21 (*Weak-Blindness*)**.** *Notre protocole* TASS *est une signature en blanc sous l'hypothèse* DDH.

Démonstration. Dans cette preuve, on utilise trois jeux de sécurité. Le simulateur $\mathcal{B}$ joue le rôle du token $\mathcal{T}$ et de l'utilisateur $\mathcal{U}$. L'adversaire $\mathcal{A}$ joue le rôle du serveur $\mathcal{S}$.

$\mathbf{G_0}$. Dans ce jeu, le protocole initial tourne sans modifications. L'algorithme Setup génère $h = g^{\alpha}$. $\mathcal{A}$ génère une clé de vérification vk et deux messages m_0 et m_1. Les trois parties du protocole signent en premier le message m_b puis le message m_{1-b}. À la fin, le simulateur $\mathcal{B}$ obtient une signature de Waters valide sur le message m_b et sur le message m_{1-b}. Il randomise la signature avec $s' \xleftarrow{\$} \mathbb{Z}_p$ pour obtenir Σ_b et Σ_{1-b}. L'avantage de $\mathcal{A}$ est donc :

$$\epsilon = \mathsf{Adv}^{bl}_{\mathcal{BT},\mathcal{A}}(k) = \Pr_{G_0}[b' = 1][b = 1] - \Pr_{G_0}[b' = 1][b = 0].$$

$\mathbf{G_1}$. Le but de ce jeu est de faire en sorte que les deux signatures générées pour les messages m_0 et m_1 ne dépendent plus de b. Pour cela, nous devons modifier le message dans la partie de l'utilisateur. En faisant cela, nous modifions les outputs de l'algorithme de chiffrement. Ce dernier vérifie la propriété de sécurité $\mathsf{IND\text{-}CPA}$ *i.e.* ces outputs sont indiscernables entre eux. Ainsi, la modification des messages ne sera pas vu par $\mathcal{A}$. Ensuite, on continue le protocole normalement. Le protocole de chiffrement utilisée est la version labellisée du chiffrement de Cramer-Shoup. Comme rappelé dans la section 6.4.1, ce chiffrement est $\mathsf{IND\text{-}PCA}$, donc $\mathsf{IND\text{-}CPA}$. Comme Enc est $\mathsf{IND\text{-}CPA}$ sous l'hypothèse DDH notre signature vérifie la *weak blindness* sous l'hypothèse DDH.

□

Théorème 22 (Signature à seuil)**.** *Notre protocole* TASS *vérifie la propriété de signature à seuil sous l'hypothèse* $\mathsf{CDH+}$.

Démonstration. On construit un challenger $\mathcal{B}$ qui joue les jeux $\mathsf{CDH+}$. Le but du serveur $\mathcal{S}$ est de réussir à trouver la clé de signature du token $\mathcal{T}$ et inversement pour ce dernier. De même que pour la preuve de résistance aux collisions, $\mathcal{B}$ reçoit le challenge $([x_1]_1, [x_1]_2, [x_2]_2)$. Le but est de trouver $[x_1x_2]_1$. Si le token ou le serveur obtiennent cette donnée, ils pourront effectuer des signatures valides l'un sans l'autre.

$\mathbf{G_0}$. Dans le jeu initial, le jeu 0, il n'y a pas de modification. On répond honnêtement en utilisant les algorithmes originaux du protocole.

$\mathbf{G_1}$. Dans ce jeu, nous effectuons les mêmes modifications que dans la preuve de la résistance aux contrefaçons. La place de l'adversaire va être prise successivement par le serveur corrompu $\mathcal{S}^*$ puis par le token corrompu $\mathcal{T}^{\times}$. Le challenger va les remplacer pour répondre à leurs requêtes. Ils ont ainsi chacun accès à $k-1$ signatures valides (les requêtes de signatures de $\mathcal{S}^*$ et de $\mathcal{T}^{\times}$ ne portent pas sur les mêmes messages.). On note par m^* le message challenge associé au serveur malveillant $\mathcal{S}^*$. De même, $m^{\times}$ est associé au token malveillant $\mathcal{T}^{\times}$.

Pour les messages challenges m^* et $m^{\times}$, on répond sur le même modèle que dans la preuve de la résistance aux collisions. On peut donc conclure en utilisant le même argument que dans cette preuve *i.e.* la résistance aux collisions de la signature asymétrique de Waters sous l'hypothèse $\mathsf{CDH+}$.

On en conclut que notre signature vérifie la propriété de signature à seuil sous l'hypothèse $\mathsf{CDH+}$

□

6.7 Conclusion

Cette partie du manuscrit était dédiée à l'étude d'un protocole de signature à trois parties. Les trois parties considérées étaient un utilisateur $\mathcal{U}$, un token $\mathcal{T}$ et un serveur $\mathcal{S}$.

Le principe de notre signature est de permettre à un utilisateur de faire signer un message par le serveur et le token sans que ces derniers ne puissent le faire l'un sans l'autre.

Afin d'établir cette signature, plusieurs briques cryptographiques sont utilisées tels que les *smooth projective hash function*, la signature de Waters et le chiffrement labellisé de Cramer-Shoup.

Notre construction est présentée dans le modèle standard sous des hypothèses classiques (CDH+ et DDH). La signature est résistante aux collisions, vérifie la propriété de *weak blindness* et la signature ne peut pas être générée par le token ou le serveur indépendamment de l'autre partie.

Chapitre 7

Conclusion

Cette thèse a permis d'étudier à la fois des protocoles de chiffrement mais également des protocoles de signatures. L'utilisation de ces deux primitives cryptographiques permet de développer des outils d'identification adaptés au monde actuel.

Tout en utilisant un chiffrement basé sur l'identité, il est possible, grâce au protocole AIBET de déterminer l'identité du destinataire du message. Ce protocole permet une utilisation dans des institutions où les informations sont soumises à une hiérarchie. D'un point de vue technique, deux instanciations d'un AIBET furent proposées en conservant les propriétés de sécurité des protocoles. Ainsi, en partant d'un IBE pré-existants et en conservant les propriétés de sécurité (sécurité sémantique et anonymat), l'identité du destinataire est traçable tout en protégeant les données échangées. La clé de chiffrement, tsk permet de vérifier si l'identité du destinataire correspond à une identité recherchée mais ne peut pas permettre le déchiffrement du message. De plus, si les identités ne sont pas identiques, aucune information ne peut être tirées de l'équation du couplage de vérification.

Les protocoles de signatures présentés au fil de cette thèse ont également des applications concrètes. Notre protocole de signature en blanc, de taille constante et en un seul tour, permet de signer efficacement des documents de transactions bancaires ou de monnaies électroniques tout en reposant sur une hypothèse de sécurité standard. Le résultat associé à cette signature, la SRC de taille constante, permet d'être utilisée, elle aussi, pour le vote électronique. Tout en étant applicable dans notre vie quotidienne, la sécurité de ces signatures repose sur des hypothèses classiques et nécessite un nombre restreint d'éléments de groupe.

La signature à base d'attributs, notée ABDVS, permet au signataire de désigner un groupe de personnes comme vérifieurs. Cette signature utilise un protocole de chiffrement à base de *Smooth Projective Hash Function*. L'utilisation de ce protocole permet d'assurer les propriétés de sécurité de notre signature ABDVS : résistance aux contrefaçons, non-transférabilité et respect de la vie privée de l'utilisateur.

La dernière signature proposée permet à trois entités de signer un message. Dans cette signature, notée TASS, les trois entités ne peuvent pas signer un document sans la présence des deux autres. Grâce à cela, les différents utilisateurs du protocole sont protégés. Comme toute signature, elle est résistante aux contrefaçons sous l'hypothèse CDH+.

Un protocole de chiffrement analogue est en cours de soumission à la conférence PKC.

Une suite possible à ces recherches est l'extension de ces protocoles aux réseaux euclidiens notamment pour le protocole de chiffrement AIBET. La signature en blanc, reposant sur une signature SPS, on peut essayer de l'améliorer encore en utilisant une signature SPS *randomisable*, ce qui n'est pas possible actuellement. On peut également chercher à développer la signature en blanc à la cryptographie post-quantique. Concernant la signature TASS, il serait intéressant de proposer un schéma dans le modèle de la composabilité universelle (*universal composability*).

Publications et interventions

Interventions

— 2019 : Présentation d'un travail de groupe sur l'étude d'un réseau de neurones dans le cadre des voitures autonomes (*Robustness evaluation for AI-based IDS against AI-based Opponent Models*) en association avec l'IRT System X lors des journées Redocs organisées par le GDR Sécurité Informatique, Gif-sur-Yvette, France.

— 2019 : Présentation de l'article *Anonymous Identity-based Encryption* à la conférence internationale ARES 2019, Canterbury, Angleterre.

— 2018 : Présentations de l'article *Anonymous Identity-based Encryption* aux journées cryptographies et codages 2018, Aussois, France.

Publications

[BBCF20] Olivier Blazy, Laura Brouilhet, Céline Chevalier, and Neals Fournaise. Round-optimal constant-size blind signatures. In Pierangela Samarati, Sabrina De Capitani di Vimercati, Mohammad S. Obaidat, and Jalel Ben-Othman, editors, *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2 : SECRYPT, Lieusaint, Paris, France, July 8-10, 2020*, pages 213–224. ScitePress, 2020.

[BBP19] Olivier Blazy, Laura Brouilhet, and Duong Hieu Phan. Anonymous identity based encryption with traceable identities. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, New York, NY, USA, 2019. Association for Computing Machinery.

En cours de soumission

1. *Attribute-based Designated Verifier Signature* avec Olivier Blazy, Emmanuel Conchon et Mathieu Klingler.
2. *Hardware security without secure hardware : How to sign with a password and a server ?*
   Olivier Blazy, Céline Chevalier, Patrick Towa, Ida Tucker et Damien Vergnaud
3. *Hardware security without secure hardware : How to encrypt with a password and a server ?*
   Olivier Blazy, Céline Chevalier, Patrick Towa, Ida Tucker et Damien Vergnaud

Table des figures

Bibliographie

[AB04] A. Atkin and Daniel Bernstein. Prime sieves using binary quadratic forms. *Math. Comput.*, 73 :1023–1030, 04 2004. 4

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, Heidelberg, December 2013. 17

[ABC+05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited : Consistency properties, relation to anonymous IBE, and extensions. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, Heidelberg, August 2005. 22

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 332–352. Springer, Heidelberg, March / April 2015. 83

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org/2005/385`. 13

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, Heidelberg, August 2009. 17

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010. 26, 30, 59

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Heidelberg, May 2004. 22, 23

[BBCF20] Olivier Blazy, Laura Brouilhet, Céline Chevalier, and Neals Fournaise. Round-optimal constant-size blind signatures. In Pierangela Samarati, Sabrina De Capitani di Vimercati, Mohammad S. Obaidat, and Jalel Ben-Othman, editors, *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2 : SECRYPT, Lieusaint, Paris, France, July 8-10, 2020*, pages 213–224. ScitePress, 2020. 7, 26, 49

[BBP19] Olivier Blazy, Laura Brouilhet, and Duong Hieu Phan. Anonymous identity based encryption with traceable identities. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, New York, NY, USA, 2019. Association for Computing Machinery. 7, 35

[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. 13, 32

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2) :156 – 189, 1988. 16

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001. 16, 31, 62

[BF03] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3) :586–615, 2003. 5, 22

[BFM90] Manuel Blum, Paul Feldman, and Silvio Micali. Proving security against chosen cyphertext attacks. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 256–268. Springer, Heidelberg, August 1990. 18

[BFPV11] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 403–422. Springer, Heidelberg, March 2011. 14, 30, 49, 53, 59, 85

[BGP19] Olivier Blazy, Paul Germouty, and Duong Hieu Phan. Downgradable identity-based encryption and applications. In *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings*, pages 44–61, 2019. 24, 37, 41

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, August 2014. 7, 31, 34, 37, 41, 42, 71, 78

[Blu81] Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *CRYPTO'81*, volume ECE Report 82-04, pages 11–15. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981. 16

[Bon98] Dan Boneh. The decision diffie-hellman problem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 48–63, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. 13

[BOY86] C. BOYD. Digital multisignatures. *Cryptography and Coding*, pages 241–246, 1986. 80

[BP13] Fabrice Benhamouda and David Pointcheval. Trapdoor smooth projective hash functions. Cryptology ePrint Archive, Report 2013/341, 2013. `http://eprint.iacr.org/2013/341`. 18

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 94–111. Springer, Heidelberg, March 2012. 17, 59

[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. 15

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer, Heidelberg, August 2006. 7, 36, 62, 64, 78

[Can01] Ran Canetti. Universally composable security : A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. 80

[CFH+07] Yang Cui, Eiichiro Fujisaki, Goichiro Hanaoka, Hideki Imai, and Rui Zhang. Formal security treatments for signatures from identity-based encryption. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 218–227. Springer, Heidelberg, November 2007. 42, 46

[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. 6, 26, 40, 48

[CLNS16] Jan Camenisch, Anja Lehmann, Gregory Neven, and Kai Samelin. Virtual smart cards : How to sign with a password and a server. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 353–371. Springer, Heidelberg, August / September 2016. 80

[Coc01a] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 360–363. Springer, Heidelberg, December 2001. 5, 22

[Coc01b] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 360–363. Springer, Heidelberg, December 2001. 31

[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. 16, 32, 83

[CS01] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. Cryptology ePrint Archive, Report 2001/085, 2001. `http://eprint.iacr.org/2001/085`. 17

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991. 48

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001. 18

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976. 4, 14, 25, 31

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013. 14, 15

[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. 32

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31 :469–472, 1985. 31

[Eve81] Shimon Even. Protocol for signing contracts. In Allen Gersho, editor, *CRYPTO'81*, volume ECE Report 82-04, pages 148–153. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981. 16

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Heidelberg, August 2006. 7, 26, 27, 55, 81

[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. 18

[FWCS12] Chun-I Fan, Chien-Nan Wu, Wei-Kuei Chen, and Wei-Zhe Sun. Attribute-based strong designated-verifier signature scheme. *J. Systems and Software*, 85 :944–959, 2012. 41, 47, 48

[GH07] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 265–282. Springer, Heidelberg, December 2007. 23

[GHPT17] Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. Identity-based Encryption from Codes with Rank Metric. In Jonathan Katz and Hovav Shacham, editors, *Crypto 2017 - Advances in Cryptology*, volume 10403 of *LNCS - Lecture Notes in Computer Science*, pages 194–224, Santa-Barbara, United States, August 2017. Springer. 63

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Heidelberg, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`. 17

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982. 20

[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. 18

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2) :281–308, April 1988. 5, 25, 26

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. 25

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, Heidelberg, December 2002. 23

[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. 10, 18, 55

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 466–481. Springer, Heidelberg, April / May 2002. 23

[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. 48

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer, Heidelberg, May 2005. 17

[KPW15] Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-preserving signatures from standard assumptions, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 275–295. Springer, Heidelberg, August 2015. 39, 48, 49, 54, 91

[KTY04] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 571–589. Springer, Heidelberg, May 2004. 6, 48, 61

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of Structures in Complexity Theory*, pages 102–111, 1993. 27

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979. 5

[MPR11] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 376–392. Springer, Heidelberg, February 2011. 27, 40, 48

[MRV15] Paz Morillo, Carla Ràfols, and Jorge L. Villar. Matrix computational assumptions in multilinear groups. Cryptology ePrint Archive, Report 2015/353, 2015. `http://eprint.iacr.org/2015/353`. 15

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989. 16

[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. 20

[OP01] Tatsuaki Okamoto and David Pointcheval. REACT : Rapid Enhanced-security Asymmetric Cryptosystem Transform. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 159–175. Springer, Heidelberg, April 2001. 21

[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. 54

[Rab79] Michael O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, January 1979. 5

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992. 20

[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. 4, 31

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal designated-verifier signatures. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 523–542. Springer, Heidelberg, November / December 2003. 29

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984. 5, 22, 31

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. `http://eprint.iacr.org/2001/112`. 15

[Sho02] Victor Shoup. OAEP reconsidered. *Journal of Cryptology*, 15(4) :223–249, 2002. 15

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005. 5, 25

[SZM04] Willy Susilo, Fangguo Zhang, and Yi Mu. Identity-based strong designated verifier signature schemes. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP 04*, volume 3108 of *LNCS*, pages 313–324. Springer, Heidelberg, July 2004. 48

[VVdB17] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham : Springer International Publishing*, 2017. 41

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005. 38, 41

Résumé

Dans cette thèse, nous utilisons des protocoles cryptographiques existants afin d'en proposer de nouveaux ou avec de nouvelles propriétés intéressantes.
Nous avons tout d'abord proposer un protocole de signature à base d'attributs à partir d'un chiffrement basé sur l'identité. La sécurité de cette construction est prouvée sous une hypothèse classique. Par la suite, nous proposons une signature en blanc en tour optimal et de taille constante grâce à la méthode de construction de Fischlin et des preuves non-interactives à divulgation nulle de connaissance. De plus, cette signature est prouvée sûre sous une hypothèse classique. En résultat annexe, nous proposons une signature sur chiffré *randomizable* de taille constante est également présentée et prouvée sous la même hypothèse.
Ensuite, nous introduisons un nouveau protocole de chiffrement basé sur l'identité (IBE) qui permet a un traceur, avec une clé associé à une identité, de filtrer tout les chiffrés envoyé à cette identité précise (et seulement celle-ci).
Finalement nous proposons un protocole de signature à trois parties prouvée sûr sous des hypothèses standards. Cette construction utilise différents outils tels que des SPHF ou la signature asymétrique de Waters.

Abstract

In this thesis, we use building blocks to propose new one or with new interesting properties.
First, we propose a attribute-based designated verifier signature thanks to an IBE. Security properties are proven under usual hypothesis. Then, we introduce our round-optimal constant-size blind signature thanks to Fischlin framework and NIZK. As a side result, we propose a constant-size signature on randomizable ciphertexts.
Then, we introduce a new IBE which allows a tracer, given a tracing key associated to an identity, to filter all the ciphertexts that are sent to this specific identity (and only those). Two applications of this protocols are proposed. We show that our modification doesn't alter the security of IBE.
Finally, we present a threshold signature between an user, a token and a server thanks to different building blocks like SPHF or assymetric Waters signature. The security of the construction is proven under regular assumptions like CDH+ or DDH.