Protocoles cryptographiques pour le respect de la vie privée

Thèse

présentée et soutenue publiquement le 4 Janvier 2022 par

Neals Fournaise

pour l'obtention du

Doctorat de l'Université de Limoges
(spécialité Informatique)

Devant le jury composé de :

Directeurs de thèse :	Olivier Blazy	PR à l'École Polytechnique
	Philippe Gaborit	PR à l'Université de Limoges
Rapporteurs :	Damien Vergnaud	PR à l'Université PSL / IUF
	Pascal Lafourcade	MCF/HDR à l'Université Clermont Auvergne
Examinateurs :	Céline Chevalier	MCF/HDR à l'Université Panthéon-Assas Paris 2
	Amandine Jambert	Ingénieure Experte à la CNIL
	Duong Hieu Phan	PR à Telecom Paris

Remerciements

Afin de conclure cette aventure, humaine plus que scientifique, qu'a été cette thèse, je tiens à remercier les personnes qui m'ont entouré, professionnellement et personnellement. Tout d'abord, je remercie mon directeur de thèse Olivier Blazy. Tu as su être présent du début jusqu'à la fin, malgré les nombreux aléas rencontrés, et m'apporter un soutien presque sans faille lorsque c'était nécessaire. Merci pour toutes les discussions, tes conseils toujours pertinents, merci d'avoir partager avec moi tes connaissances et ta vision du fabuleux domaine qu'est la recherche en cryptographie.

Je remercie sincèrement Philippe Gaborit, aussi mon directeur de thèse, avec qui j'ai eu la joie d'apprendre tout au long de mon cursus universitaire et de collaborer scientifiquement.

Je remercie également Duong Hieu Phan pour sa direction pendant ma première année de thèse, ses conseils, ainsi que pour son accompagnement de qualité durant mes deux années de master.

Sans le travail des rapporteurs, je n'aurais pas la possibilité de soutenir. Un grand merci à Pascal Lafourcade et à Damien Vergnaud d'avoir accepté cette tâche et d'avoir passé du temps sur ce manuscrit afin de m'en donner un juste retour.

Je tiens aussi à remercier Céline Chevalier avec qui j'ai eu le plaisir de travailler à plusieurs reprises durant ma thèse. Merci à Amandine Jambert de me faire l'honneur d'être dans mon jury de soutenance.

La majeure partie de mon parcours universitaire s'est déroulé à Limoges et j'ai donc eu la chance d'être à la fois étudiant (pour certains) et collègue des membres de l'équipe du département Informatique (et pas que). Merci à Damien Sauveron, Christophe Clavier, Emmanuel Conchon, Pierre-François Bonnefoi, Olivier Terraz, Benoît Crespin, Tristan Vaccon, Guillaume Gilet, Maxime Maria. Merci également à Débora Thomas et Sophie Queille pour leur précieuse aide administrative.

Je ne sais pas s'il faut vous remercier ou vous reprocher votre capacité à me distraire continuellement : Mathieu, Nicolas, Maxime, Théo, Laura. Ces années avec vous ont été franchement marrantes et agréables.

Merci à mes amis, Sarah, Yaniv, Camille, d'avoir été là quand il le fallait mais aussi quand il ne le fallait pas.

Table des matières

Table des matières

Chapitre 1

Introduction

La cryptographie est née d'un besoin de confidentialité que l'on peut retrouver dès l'antiquité. En effet, il est aisé d'imaginer que dissimuler de l'information militaire ou politique puisse avoir une importance capitale sur l'avenir d'une guerre ou d'un gouvernement. Ainsi, des codes secrets, des méthodes de chiffrement, ont été créés très tôt et s'en est suivie une longue bataille entre cryptographes et cryptanalystes ; les uns désirant cacher les messages, les autres les révéler en exploitant les faiblesses potentielles des chiffrements. Si cette dualité entre attaquants et défenseurs est toujours essentielle dans l'étude de la cryptologie moderne, elle n'est plus nécessairement centrale et les objectifs historiques ont laissé place à une foule d'applications diverses et variées.

L'évolution des sciences mathématiques, informatiques et physiques a permis un développement accéléré de la discipline depuis l'après-guerre. Ses fondements mêmes ont graduellement changés pour devenir une science à part, un sous-domaine des mathématiques ; une approche extrêmement rigoureuse étant de plus en plus indispensable au fur et à mesure que les techniques cryptographiques se complexifient. Son théâtre d'opération n'est aussi plus uniquement militaire et politique ; nous retrouvons les outils cryptologiques dans nombre de cas d'utilisation commerciale, scientifique, et technologique.

L'avènement des nouvelles technologies de communication a permis à la cryptographie d'avoir un essor considérable et d'être désormais au cœur de tous nos échanges, qu'ils soient entre individus ou machines. Le monde d'aujourd'hui est informatique, connecté, complexe ; Internet a bouleversé nos modes de vie et quasiment aucune de nos activités ne peut se passer de la cryptographie, invisible mais essentielle. Les systèmes bancaires, les systèmes de paiement électroniques, les services de messageries (instantanées ou non), les réseaux, l'informatique en nuage, etc., ont tous besoin d'être sécurisés et encore une fois, les outils cryptographiques en sont un élément essentiel.

1.1 Les bases

Pour présenter la cryptographie et ses enjeux actuels, il est souvent bon de rappeler comment tout a commencé afin d'amener petit à petit certains grands principes qui

subsistent ou non aujourd'hui.

1.1.1 Le chiffrement à clé privée

C'est avec le chiffrement que tout commence : l'idée est de pouvoir rendre inintelligible de l'information (un message, un document, etc.) à une tierce personne indiscrète. Ainsi, un schéma de chiffrement comporte trois éléments :

— *La génération de clés* Gen : elle permet de créer une clé k tirée aléatoirement à partir d'un ensemble donné,

— *Le chiffrement* Enc : chiffre un message m avec une clé k pour produire un chiffré c que l'on note $c = \mathsf{Enc}_k(m)$,

— *Le déchiffrement* Dec : à partir d'un chiffré c et d'une clé k, il permet de récupérer m. Nous avons la relation $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$.

Ainsi pour communiquer secrètement, deux personnes doivent préalablement avoir généré puis partagé une même clé privée k. Alors, l'utilisation des algorithmes de chiffrement et de déchiffrement leur assurera une correspondance confidentielle du moment que personne n'entre en possession de la clé k. En possession de k, Alice peut utiliser $\mathsf{Enc}_k(m)$ pour produire un chiffré c qu'elle communiquera à Bob. À son tour, il utilisera cette même clé k avec $\mathsf{Dec}_k(c)$ afin de récupérer le message initial m. Cela introduit la première application toute naturelle du chiffrement, illustrée dans la figure 1.1.

Une deuxième utilisation évidente du chiffrement consiste à garder pour soi la clé k et consigner des documents secrets (un journal intime, le plan d'une carte au trésor, d'une invention, etc.) si l'on n'a pas la garantie que le support de l'information ne tombe pas en de mauvaises mains.

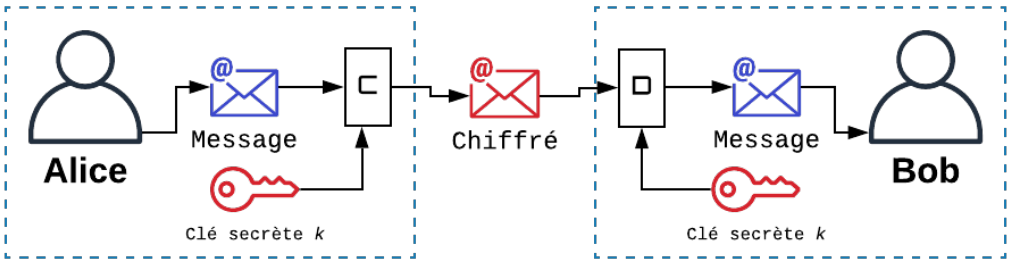

FIGURE 1.1 – Le chiffrement symétrique. Ici, la boîte C représente le chiffrement et la boîte D le déchiffrement.

1.1.2 Chiffrements historiques

Les chiffrements historiques, même s'ils ne présentent aucun intérêt autre que pédagogique à l'heure actuelle, ont l'avantage d'être relativement simples dans leur conception

et permettent d'introduire des notions essentielles.

Un des plus anciens, le chiffrement de César consiste à décaler chacune des lettres de trois positions dans l'alphabet, créant ainsi une *substitution*. Ainsi, A devient d, B devient e, C devient f et ainsi de suite. Si l'on peut penser qu'un chiffré, par exemple $erqmrxu$ pour $BONJOUR$, semble secret, il n'en est rien. En admettant que l'on ne connaisse pas le décalage (trois dans le cas présent), il suffirait de tester seulement les 26 possibilités pour récupérer le message original. De plus, nous pouvons voir une répétition dans le chiffré : pour chaque lettre O du message original, nous avons la lettre r. Cela met en lumière deux premiers points qui ont permis à de futurs cryptographes de mieux concevoir leur chiffre : la nécessité d'avoir une clé, ici le décalage, plus grande ainsi qu'un système amenant moins de structure.

Un chiffrement historique un peu plus avancé a été décrit par Blaise de Vigenère, un diplomate, dans son *traité des chiffres* en 1586. Désormais, la clé n'est plus un seul nombre compris entre 0 et 25 mais un mot de taille arbitraire t où chaque lettre correspond à un décalage. Ainsi, la première lettre de la clé donne le décalage de la première lettre du message, la deuxième lettre donne le décalage de la deuxième lettre et ainsi de suite. Lorsqu'il n'y a plus assez de lettres dans la clé, il suffit de reprendre la première de celle-ci. Un exemple : si la clé est **cafe** et le message $BONJOUR$, B est décalé de 2 par **c**, O est décalé de 0 par **a**, N est décalé de 5 par **f**, etc. ; lorsque l'on arrive au deuxième O, nous décalons de nouveau de 2 par **c**, U est décalé de 0 par **a**, etc. Cette approche permet d'une part d'augmenter le nombre de clés, en fonction de la taille t des mots, que l'on peut exprimer par 25^t dans l'idéal. La *substitution* n'est également plus nécessairement mono-alphabétique comme le montre l'exemple donné à l'instant. Pourtant, un tel chiffrement est bien plus vulnérable que l'on ne pourrait le croire et des outils de cryptanalyse, comme l'analyse fréquentielle ou l'indice de coïncidence, permettent d'en venir aisément à bout, abaissant la difficulté de récupérer la clé bien en dessous de la difficulté de tester 25^t clés possibles !

1.1.3 Principe de Kerckhoffs

Une idée survient alors : pourquoi ne pas cacher les rouages du système de chiffrement afin d'éviter la découverte de ses faiblesses ? Cela s'appelle la *sécurité par l'obscurité* et si elle peut sembler intéressante de prime abord, elle s'effondrera à la première divulgation des algorithmes. Ainsi, Auguste Kerckhoffs, cryptologue militaire néerlandais, énonce à la fin du XIXème siècle, un principe en six points [1] :

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 entre les mains de l'ennemi ;
3. La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants ;
4. Il faut qu'il soit applicable à la correspondance télégraphique ;

1. Repris depuis `https://fr.wikipedia.org/wiki/Principe_de_Kerckhoffs`.

5. Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes ;
6. Enfin, 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. »

Si certains points peuvent paraître relativement désuet, le second apparaît comme fondamental et est encore suivi aujourd'hui par les concepteurs de systèmes cryptographiques : *la sécurité d'un système ne doit reposer que sur le secret de la clé.*

1.1.4 La machine Enigma

Une première automatisation moderne du chiffrement est introduite avec la machine Enigma développée, à l'origine dans un but commercial, par l'ingénieur allemand Arthur Scherbius dans les années 1920. Le principe de cette machine électromécanique portative est fondé sur des courants électriques passant dans les différents composants, partant d'un clavier jusqu'à un tableau de lampes indiquant la lettre de chiffrement ou déchiffrement. Les différents réglages des rotors et autres éléments de la machine créent alors un nombre de possibilités relativement énorme (de l'ordre de 10^{20}) et donc impossible à attaquer par la force brute.

Son utilisation par l'armée Allemande et ses alliés de l'axe durant la Seconde Guerre mondiale a contribué au fait que nombre de spécialistes en cryptanalyse se sont attaqués au problème. Ces travaux ont permis le développement de « bombes cryptographiques », dispositifs électromécaniques permettant le déchiffrement, aidant prodigieusement les Alliés à remporter certaines batailles.

Cependant, c'est à partir de la publication de deux articles de l'américain Claude Shannon, *A Mathematical Theory of Cryptography* et *A Mathematical Theory of Communication*, que la cryptographie prend une tournure véritablement scientifique et moderne. Il y introduit des concepts encore présents aujourd'hui tels que le « bit », l'entropie de Shannon mais aussi celle de la sécurité parfaite et de la sécurité calculatoire.

1.2 Cryptographie moderne

À partir des années 70, on voit apparaître énormément de nouveaux concepts cryptographiques, sans que l'on ait forcément de solution efficace, tels que le chiffrement basé sur l'identité, les calculs multi-utilisateurs, les signatures aveugles, les preuves à divulgation nulle de connaissance, les fonctions de hachage et la liste peut être très longue. Pour beaucoup, nous retrouvons ces outils encore maintenant dans des versions actualisées, plus sécurisées, plus efficaces, dont la sécurité est également plus assurée. D'ailleurs, la sécurité d'un schéma cryptographique, son évaluation ainsi que sa compréhension, deviennent des sous-domaines à part de la discipline. Désormais, la cryptographie est vouée à être utilisée au travers de l'informatique et la sécurité s'exprime en fonction des capacités des ordinateurs contemporains.

1.2.1 Cryptographie symétrique

Historiquement, c'est la cryptographie dite symétrique qui existait et évidemment celle-ci perdure et évolue. Ainsi, nous pouvons citer l'algorithme DES comme un exemple de chiffrement symétrique moderne, utilisé avec une clé de 64 bits dont seulement 56 constituent la sécurité de la clé. En effet, à ce moment là, 56 bits de sécurité, si l'on exclut les faiblesses potentielles de l'algorithme, semblaient suffisants au regard des capacités calculatoires de l'époque. Dans le même temps, des méthodes telles que le double ou triple chiffrement DES permettent aussi de multiplier directement la complexité calculatoire nécessaire afin de parer d'éventuels progrès informatiques. Cependant, assez rapidement, des méthodes d'analyse émergent telles que la « cryptanalyse différentielle » ou la « cryptanalyse linéaire » permettant d'abaisser théoriquement la sécurité annoncée.

La cryptographie symétrique porte son nom, à l'origine, de l'existence d'une même clé secrète, partagée entre l'émetteur et le destinataire. Néanmoins, nous incluons dans cette grande famille des outils différents du chiffrement du fait de leur proximité théorique et conceptuelle. L'exemple le plus notable est illustré par les fonctions de hachage qui permettent de générer une empreinte de taille fixe à partir d'une entrée de taille arbitraire. L'application directe étant la préservation de l'intégrité dans le sens où à une entrée correspond une seule empreinte : un seul bit modifié sur l'entrée engendre une empreinte totalement distincte. Ces fonctions peuvent être pensées comme des fonctions à sens unique pour lesquelles il est aisé de calculer l'empreinte à partir d'une entrée mais il est difficile de récupérer l'entrée à partir d'une empreinte. Un des autres aspects essentiels de la sécurité de ces fonctions étant la résistance à la collision : il doit être difficile de trouver deux entrées ayant la même empreinte. Ainsi, l'intégrité d'un document peut être vérifiée aisément en calculant l'empreinte et en la comparant à une empreinte de référence fournie par l'émetteur du document. La parenté avec les schémas de chiffrement symétrique s'explique par l'emploi des mêmes opérations élémentaires sur les bits (addition modulaire, décalage arithmétique, opérations logiques, etc.) qui conduit à employer des techniques d'analyse similaires pour les attaquer.

1.2.2 Cryptographie asymétrique moderne

Jusqu'à présent, les outils cryptographiques, les chiffrements dits « symétriques », présentés reposent sur l'hypothèse d'une clé au préalable partagée entre les utilisateurs. Comme les applications étaient principalement militaires, politiques ou diplomatiques, cet échange de clé se faisait de façon très concrète grâce à des rencontres dans un lieu physique où circulaient des clés imprimées sur du papier et convenues à l'avance. Par exemple, pour sécuriser les communications du « téléphone rouge », ligne directe entre les États-Unis et Moscou durant la guerre froide, les clés étaient directement échangées en passant par les ambassades et transportées par avion. Cependant, cela rendait l'utilisation du chiffrement couteuse et peu adaptable, une rencontre étant toujours nécessaire avant.

Échange de clé

Fort heureusement, deux chercheurs américains, Withfield Diffie et Martin Hellman, publient en 1976 un article intitulé *New Directions in Cryptography* dans lequel ils proposent une solution d'échange de clé reposant sur une asymétrie observée à la fois dans la vie réelle mais aussi dans les mathématiques. Notons par exemple le problème de la factorisation où il est aisé de multiplier deux grands premiers mais difficile de factoriser le produit. Ainsi, l'échange de clé Diffie-Hellman est né, reposant sur l'observation que dans certains groupes, le problème du logarithme discret (LD) est difficile, celui-ci demandant de récupérer x à partir de g^x. Dans un groupe multiplicatif d'ordre p, p premier, g étant une racine primitive, ce problème ne possède pas de résolution efficace pour une certaine taille de p.

Partant de cela, l'échange de clé de Diffie-Hellman, illustré dans la figure 1.2, consiste à générer pour Alice un grand nombre a, un scalaire, compris entre 0 et $p-1$, de calculer g^a et de l'envoyer à Bob. Il fait de même avec un nombre b et envoie g^b à Alice. Les deux parties peuvent désormais calculer un secret commun

$$(g^a)^b = (g^b)^a = g^{ab}$$

que des attaquants ne peuvent reconstituer qu'en résolvant un problème appelé le problème Diffie-Hellman calculatoire (« Computationial Diffie-Hellman problem », CDH). Celui-ci demande, étant donné un tuple (g, g^x, g^y), de trouver g^{xy}. Heureusement, il existe des groupes dans lesquels ce problème est difficile ce qui autorise l'utilisation de façon sécurisée du protocole Diffie-Hellman. De plus, ce problème est directement lié à celui du logarithme discret étant donné qu'il est aisé de remarquer que si l'on résout le problème LD sur g^x ou g^y, il est alors facile de calculer le terme g^{xy}.

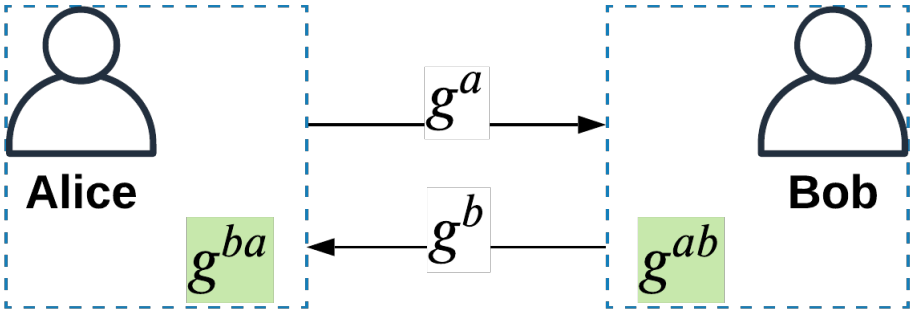

FIGURE 1.2 – L'échange de clé

Cependant, outre le souci de traduire un élément de groupe en une chaîne de bits utilisables comme clé secrète dans un algorithme de chiffrement symétrique (g^{ab} dans

notre protocole est un élément de groupe), il faut également s'assurer de l'authenticité des messages envoyés. En effet, la version naïve présentée souffre d'une grosse faiblesse et une attaque de type « man-in-the-middle » se révèle très efficace si Alice et Bob n'utilisent pas un canal de communication authentifié (mais non nécessairement confidentiel). L'authentification peut être apportée, dans le cas d'un échange de clé, grâce à un algorithme de signature numérique ou par un mot de passe préalablement partagé, par exemple.

De cette façon, un échange de clé, dans la pratique, est toujours authentifié pour garantir une sécurité suffisante. Les moyens d'authentifications sont variés et peuvent être des clés cryptographiques secrètes ou publiques (forte entropie), des mots de passe (faible entropie) ou encore des *credentials*, soient une forme d'identification attestant une appartenance, une autorisation ou autre. Nous parlons d'AKE, pour *Authenticated Key Exchange*, dans le premier cas, de PAKE, *Password-Authenticated Key Exchange*, pour le second cas et de CAKE, *Credentials-Authenticated Key Exchange*, dans le dernier cas. Dans l'article [BBC+13a], les auteurs généralisent ces primitives grâce au LAKE, *Language-Authenticated Key Exchange*. L'échange de clé est alors authentifié par la satisfation d'une relation algébrique pouvant revêtir plusieurs formes, la relation determinant le type d'échange de clé. Par exemple, pour le PAKE, la relation à satisfaire et une égalité dans les mots de passe employés par les deux utilisateurs cherchant à partager une clé. Un type d'échange de clé nous intéressant particulièrement, une construction étant donnée dans le chapitre 5, est la « Poignée de main secrète » ou *Secret Handshake*.

Ce protocole, introduit dans [BDS+03a], est nommé d'après les poignées de main utilisées dans des sociétés secrètes afin de s'authentifier en tant que membre. L'idée est que seuls les membres de la dite société vont savoir reconnaître la dite poignée de main qui aura l'air tout à fait conventionnelle à des yeux non-avertis. Dans un *Secret Handshake* cryptographique, l'échange de clé ne sera un succès qu'à la condition que les deux utilisateurs du protocole appartiennent à la même organisation. Deux propriétés sont essentielles pour cet algorithme : il ne doit pas être possible de distinguer, depuis l'extérieur, le succès ou l'échec de l'échange de clé et il ne doit pas être possible d'apprendre la moindre information sur l'organisation à laquelle les utilisateurs appartiennent. Le protocole est illustré par la figure 1.3 et nous montrerons plus tard comment l'obtenir à partir d'autres primitives cryptographiques.

Chiffrement

Une autre application de la cryptographie asymétrique est bien évidemment le chiffrement. En 1977, Rivest, Shamir et Adleman proposent l'algorithme bien connu RSA où cette fois-ci, l'information publiquement disponible, une clé, permet de chiffrer des messages pour le possesseur de la clé privée associée. La sécurité de cet algorithme repose sur la difficulté de factoriser deux grands entiers premiers. Les différentes étapes de génération d'une clé sont :

— Il faut donc générer un module N, produit de deux grands premiers p et q, $N = pq$.

— Ensuite, un exposant e est choisi, il doit être premier avec $\Phi(N) = (p-1)(q-1)$,

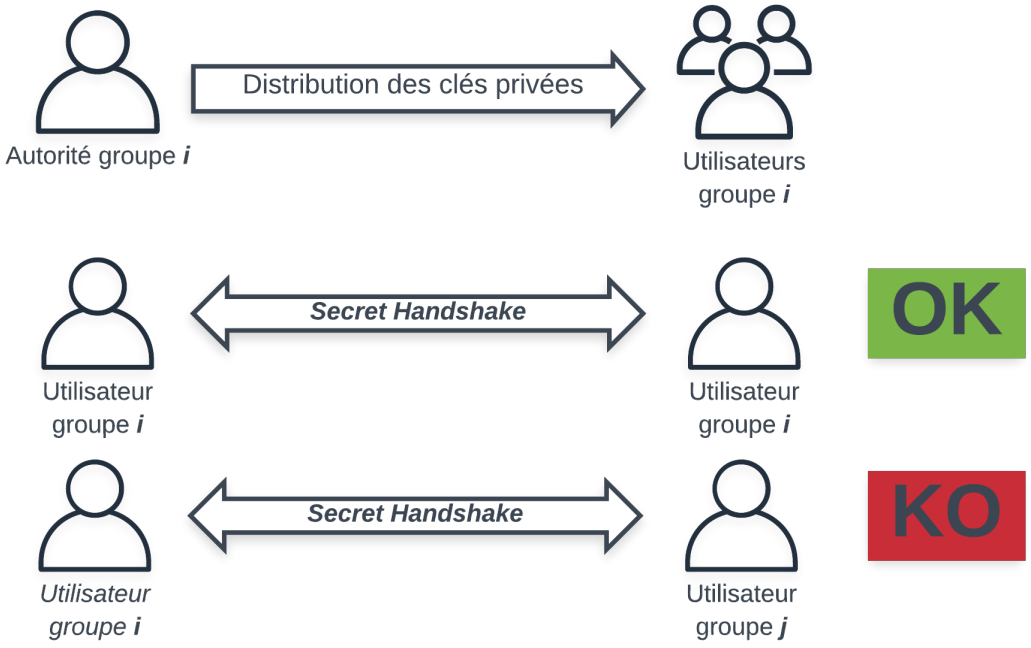

FIGURE 1.3 – La poignée de main secrète

usuellement 65537 est choisi car il est court et possède un faible nombre de bits à 1.

— Alors, il faut calculer l'inverse modulaire d de e, tel que $ed \equiv 1 \pmod{N}$, à l'aide de l'algorithme d'Euclide étendu.

La clé publique permettant à tout le monde de chiffrer sera (N, e) et la clé privée d'Alice, lui permettant, à elle seule, de déchiffrer sera (N, e, d).

Pour chiffrer un message, une suite de bits, il faut la convertir en un grand entier m. Une contrainte est que pour ne pas perdre d'information, il faut que ce grand entier converti soit inférieur au module N qui sera utilisé. Le chiffrement se fait alors en calculant

$$m^e \equiv c \pmod{N}$$

produisant un chiffré c. Le déchiffrement, quant à lui, nécessite la clé privée d et se fait comme suit

$$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m \pmod{N}$$

pour récupérer le message m. Le petit théorème de Fermat permet de prouver la validité du déchiffrement. Encore une fois, cette version de l'algorithme présentée est

naïve et manque de plusieurs éléments pour qu'une sécurité concrète soit garantie mais il pose des bases qui ont révolutionné le domaine de la cryptographie asymétrique.

Étant donnée la nature des opérations mathématiques utilisées dans l'algorithme RSA, celui-ci est relativement lent et ainsi, s'il est utilisé pour chiffrer, il vaut mieux chiffrer le moins possible. En pratique, nous pouvons retrouver RSA en tant qu'algorithme permettant de transmettre une clé secrète d'un algorithme de chiffrement symétrique, lui efficace. Cela devient alors relativement équivalent à un échange de clé comme celui de Diffie-Hellman présenté plus tôt. Un exemple de chiffrement asymétrique est donné dans la figure 1.4 dans lequel Alice enverra un message chiffré à l'aide de la clé publique de Bob. Dans l'illustration, la clé publique est récupérée sur un serveur public, appelé **PKI** pour *Public Key Infrastructure*, servant de lieu de stockage et d'échange des clés publiques. Cela permet de ne pas avoir besoin de communiquer une première fois avec Bob tout en s'assurant, si l'on fait confiance à la **PKI**, de chiffrer le message pour le bon destinataire.

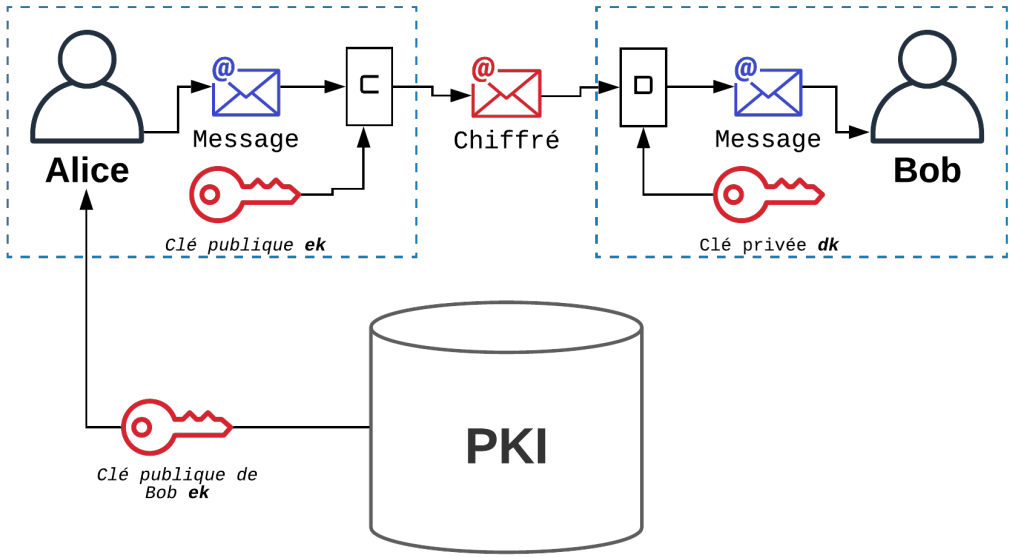

FIGURE 1.4 – Le chiffrement à clé publique

Signature numérique

Une remarque intéressante sur l'algorithme RSA est sa relation étroite avec une autre primitive cryptographique intéressante. En effet, RSA a une dualité « chiffrement-signature » qui fait que si l'on chiffre un message avec la clé privée d, celui-ci peut être déchiffré par tout le monde possédant la clé publique e. Si cela n'a pas d'utilité pour la confidentialité, nous pouvons y voir les prémisses d'un outil extrêmement utile : la

signature numérique. Celle-ci donne, en effet, l'assurance que le message vient bien du possesseur de la clé privée : c'est l'authentification ; une autre propriété intéressante étant la garantie de l'intégrité du document : il est possible de vérifier que le document signé et authentifié n'a pas été altéré.

Ainsi, pour obtenir une signature s sur un message m, il suffit de l'élever à la puissance d modulo N. Toujours avec la même relation mathématique que pour le chiffrement, une exponentiation modulaire avec la clé publique e permet de récupérer seulement m :

$$s^e \equiv (m^d)^e \equiv m \pmod{N}.$$

La difficulté de résoudre le problème RSA (ou plus fort encore, la factorisation) donne une garantie que la signature est valide et sera vérifiable avec e uniquement si elle a été produite avec la bonne clé privée d associée par rapport au module N. La propriété d'authentification est alors satisfaite. Encore une fois, cet algorithme de signature, présenté tel quel, est une version naïve qui dans la pratique ne sera pas suffisamment sécurisée mais reste un très bon exemple.

De la même façon que l'échange de clé s'est vu ajouté des propriétés, des signatures numériques existent sous une multitude de formes. Au delà de la simple signature, ont été proposés des schémas de signature de groupe (tout un groupe ou une partie doit signer), de signature d'anneau (un membre d'un « cercle » signe pour le compte des autres), de signature aveugle. Cette dernière nous intéresse spécifiquement car nous présentons dans un chapitre ultérieur une construction ayant aboutit sur une publication dans une conférence.

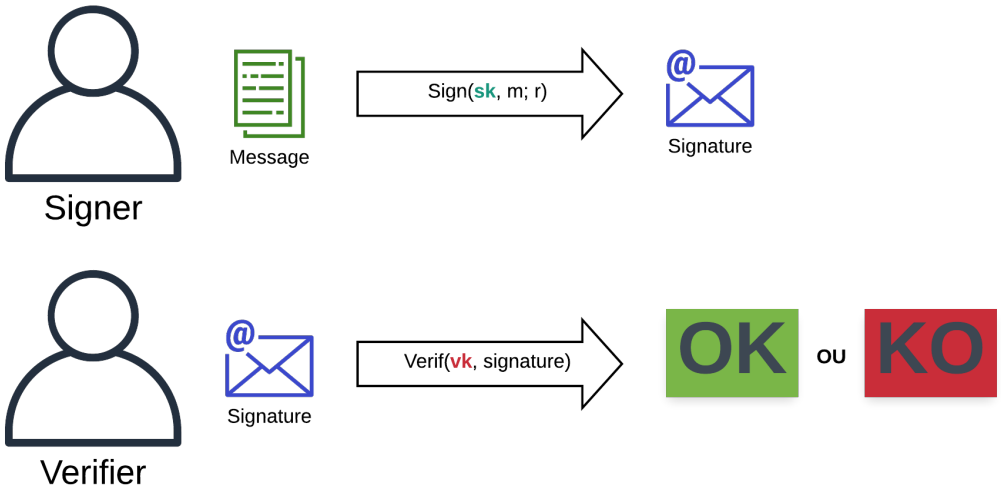

FIGURE 1.5 – La signature numérique

La signature aveugle ou *Blind Signature* consiste à transformer l'algorithme de signa-

ture classique en un protocole interactif entre un utilisateur et un serveur. L'utilisateur possède un message qu'il souhaite faire signer par le serveur tout en ne lui révélant pas le contenu de celui-ci. La signature RSA peut aisément être transformée en une signature aveugle. Plutôt que de faire signer m directement, il est possible de tirer aléatoirement un élément r qui permettra de masquer le message originel et le faire signer ainsi.

— L'utilisateur génère un message $m' \equiv mr^e \pmod{N}$ qu'il transmet au serveur signataire. Grâce à la valeur aléatoire r, aucune information sur m n'est récupérable.

— Celui-ci produit une signature $s' \equiv (m')^d \pmod{N}$ sur m'.

— Pour récupérer une signature s sur m, l'utilisateur calcule $s \equiv s'\dot{r}^{-1} \equiv m^d r^{ed} r^{-1} \equiv m^d \pmod{N}$.

Outre la résistance aux signatures contrefaites, la nouvelle propriété relative à la sécurité d'un tel schéma est celle de la « blindness » qui stipule que le signataire ne doit pas pouvoir apprendre quoique ce soit, aucune information, sur le message qu'il signe. Deux applications relativement intuitives peuvent faire usage des signatures aveugles.

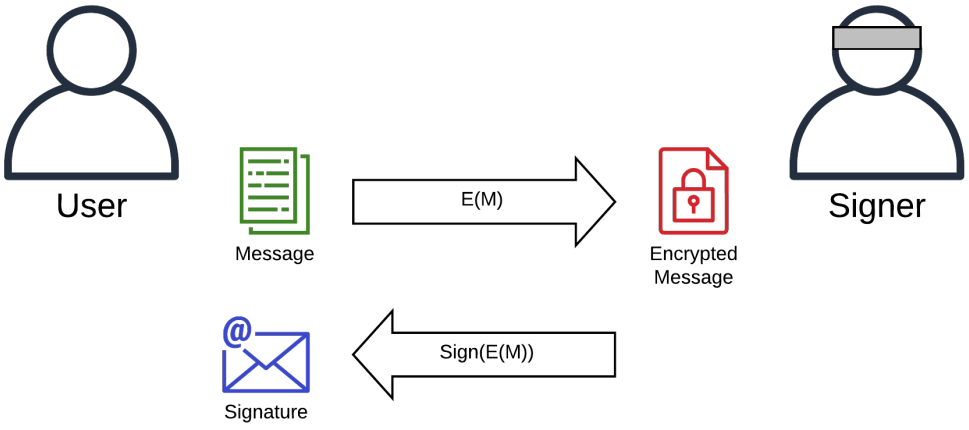

FIGURE 1.6 – La signature aveugle

La première est le vote électronique. En effet, chaque participant de l'élection possède un message, un vote, et celui-ci doit rester secret, afin d'éviter tout problème de coercition ou de corruption, tout en étant authentifié par une autorité signataire. La propriété de « blindness » permet ainsi de faire valider chacun des votes anonymement car aucune information ne sera apprise par l'autorité.

Une deuxième application peut être la monnaie électronique anonyme. En effet, à la suite de la découverte de la signature aveugle, le chercheur américain David Chaum invente un système appelé *eCash* dans lequel l'argent est stocké sous forme numérique et les paiements et transferts sont signés à l'aveugle par l'autorité centrale, la banque,

permettant ainsi d'anonymiser les transactions. Ce type de concept a depuis été repris par certaines cryptomonnaies telles que *Monero* ou encore *zCash* en utilisant des techniques cryptographiques plus modernes.

Calcul multi-utilisateurs

Comme nous venons de le voir, l'échange de clé, le chiffrement et la signature numérique sont les primitives de base de la cryptographie asymétrique. Si le chiffrement permet de garantir la confidentialité, l'échange de clé garantit l'authentification, la signature, elle, permet de garantir l'intégrité et la non-répudiation. Toutefois, il est possible d'offrir d'autres fonctionnalités aux utilisateurs des différents outils cryptographiques. Le *Secret Handshake* ou la *Blind Signature* sont, en effet, des primitives de base, auxquelles nous avons rajouté des propriétés permettant de nouveaux usages (authentification avec anonymat, vote électronique, monnaie anonyme, etc.) directement en lien avec le respect de la vie privée et de l'anonymat. De la même façon, nous retrouvons des protocoles tels que celui de Yao, introduit dans [Yao82a], dans lequel il décrit deux millionnaires désirant comparer leur fortune. À l'issue de l'exécution, la seule information connue est l'identité du plus riche des deux millionnaires mais pas les montants soumis : c'est le début du calcul multi-utilisateurs sécurisés ou *Secure Multi-Party Computation*, souvent abrégé MPC. Si le protocole de Yao porte, à l'origine, sur l'évaluation d'une relation d'inégalité entre deux nombres, des travaux ultérieurs ont permis de généraliser le MPC à presque n'importe quelle fonction modélisée à l'aide de circuits électroniques. De plus, le MPC ne porte pas nécessairement sur l'évaluation de circuits électroniques de manière sécurisée pour deux utilisateurs mais peut s'étendre à un nombre arbitraire de participant, moyennant des coûts supplémentaires, comme illustré dans la figure 1.7.

D'un point de vue technique, une primitive cryptographique a été montrée essentielle, voire complète (voir [Kil88]), pour effectuer du MPC : l'*Oblivious Transfer* (OT). Dans un protocole d'OT, il y a deux utilisateurs, l'envoyeur, possédant deux messages m_0, m_1, et le receveur, possédant un bit b, interagissent. À la fin de l'exécution de celui-ci, le receveur récupère le message m_b tandis que l'envoyeur n'apprend aucune information sur le bit b. Si la façon de faire des calculs multi-utilisateurs sécurisés à partir de l'OT dépasse le cadre de cette thèse, nous reviendrons dans une partie sur comment construire une telle primitive.

1.2.3 Post quantique

L'algorithme de Peter Shor, découvert en 1994, est venu ébranler le monde de la cryptographie asymétrique en projetant une menace potentielle pour la sécurité des cryptosystèmes reposant sur la factorisation ou le logarithme discret. En effet, ces problèmes considérés difficiles à résoudre pour des ordinateurs conventionnels deviennent résolubles en un temps relativement court, polynomial, si des ordinateurs quantiques de taille suffisante voient le jour. L'essor récent dans la recherche et le développement de tels ordinateurs fait planer un risque énorme pour la sécurité et la confidentialité des communications numériques car la majorité des algorithmes utilisés reposent justement

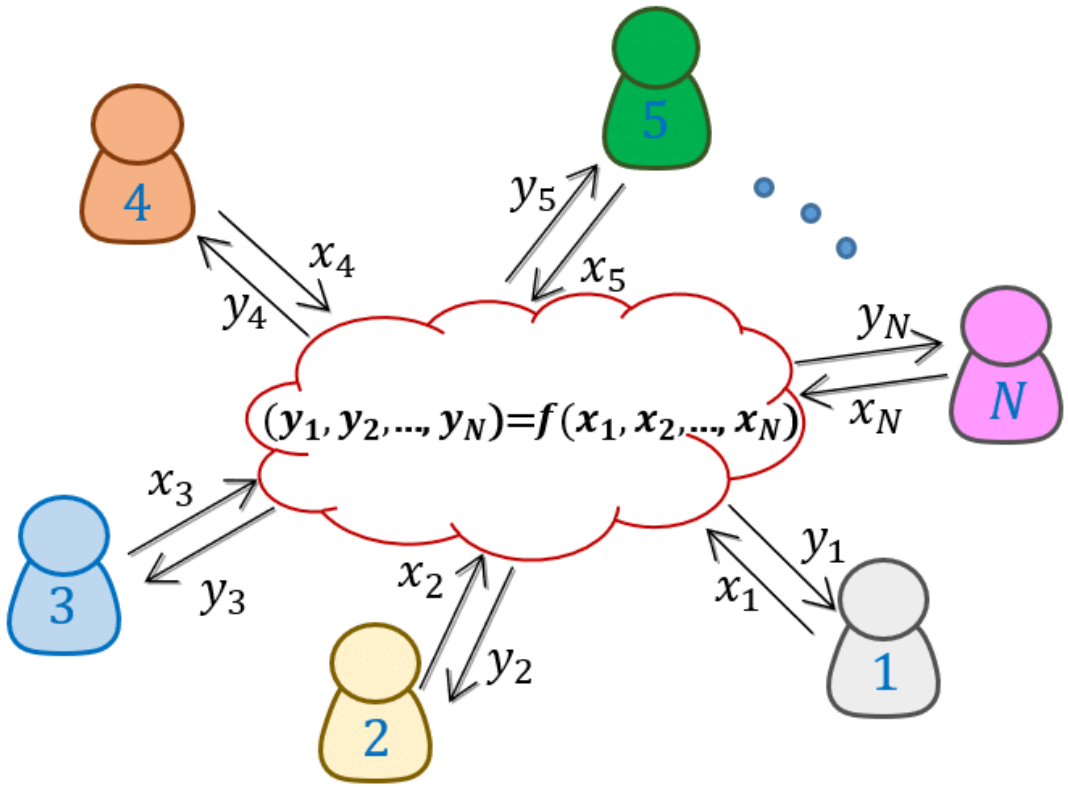

FIGURE 1.7 – Le calcul multi-utilisateurs

sur la factorisation ou le logarithme discret. À ce jour, si nous ne pouvons réellement prévoir quand et si ces machines quantiques seront suffisamment abouties, nous sommes obligés d'anticiper et de remettre en question certains fondements de la cryptographie.

Ainsi, certaines branches de la cryptographie asymétrique ont reçu un véritable élan ces dernières années et deviennent plus étudiées du fait de leur résistance à l'algorithme de Shor bien souvent au prix d'une efficacité moindre en terme de coût de calcul, de mémoire et d'une difficulté plus grande à être mises en place. Elles constituent donc la cryptographie « post quantique » ou encore « résistant au quantique ». Citons la cryptographie basée sur les réseaux euclidiens, la cryptographie basée sur les codes correcteurs d'erreurs, la cryptographie basée sur les systèmes multi-variés, la cryptographie basée sur les isogénies, la cryptographie basée sur les fonctions de hachage et encore plusieurs autres comme alternatives prometteuses à la cryptographie à clé publique plus « traditionnelles ». À noter également que la cryptographie symétrique ne se retrouve pas affectée de la même façon par la promesse de l'ordinateur quantique : l'algorithme de recherche quantique de Grover ne casse pas de la même façon leur sécurité. Un simple doublement de la taille des clés ou des problèmes suffit à pallier au problème.

1.3 Contributions

Le déroulement de la thèse n'a, pour ainsi dire, pas été linéaire. En effet, les premiers mois de recherche ont été concentrés autour d'un sujet qui ne figurera aucunement dans le présent manuscrit : les protocoles de *blockchain* et leurs relations avec la cryptographie plus « classique ». Malheureusement, toute l'étude bibliographique et les esquisses de publication n'ont pas abouties.

Ainsi, nous avons décidé de revenir à des problématiques plus courantes dans le domaine de la cryptographie asymétrique. Alors, les recherches ont été portées sur la conception de primitives cryptographiques existantes en cherchant à travailler sur plusieurs axes : l'amélioration de l'efficacité (calculs, communication, flux), les relations entres primitives elles-mêmes ainsi que leur sécurité (modèles de sécurité, preuves, hypothèses sous-jacentes).

Les travaux effectués durant la thèse ont abouti sur quatre articles, présentés dans l'ordre chronologique :

— *Efficient and Round-Optimal Secret Handshake* avec Olivier Blazy, Céline Chevalier, Emmanuel Conchon, Mathieu Klingler et Olivier Levillain, en cours de soumission.

  Dans cet article, nous présentons un protocole de *Secret Handshake*, brièvement décrit dans la section 1.2.2. Plusieurs variantes du protocole sont données et ont toutes la particularité d'être optimale en nombre de tours : un seul flux d'un utilisateur vers l'autre suffit à conclure l'authentification. De plus, elles sont efficaces d'un point de vue des coûts de communication, évaluées par le nombre d'éléments échangés mais également d'un point de vue des coûts calculatoires.

  Nos constructions ont été obtenues à partir de trois primitives cryptographiques : le chiffrement asymétrique, la signature numérique préservant la structure et un type de fonction de hachage particulière, la *Smooth Projective Hash Function*. Les différentes versions du *Secret Handshake* varient par les instantiations des différentes primitives et le respect ou non de la propriété d'*unlinkability*, le fait de pouvoir lier à un même utilisateur, sans avoir d'information sur lui, différentes exécutions du protocole. De tout cela découlent également les hypothèses de difficulté liées à la sécurité sous lesquelles elles sont prouvées.

  En outre, nous avons implémenté les différents algorithmes afin d'en évaluer les performances concrètes et montrons qu'une utilisation pratique est tout à fait envisageable sans pour autant sacrifier les performances. Par exemple, une utilisation avec TLS entraînerait un coût, au maximum, un peu plus de deux fois supérieur.

— *Round-Optimal Constant-Size Blind Signatures* avec Olivier Blazy, Laura Brouilhet et Céline Chevalier, publié.

  Dans ce travail, nous donnons, à notre connaissance, la première construction d'une signature aveugle, *Blind Signature*, brièvement présentée dans la section 1.2.2, qui est à la fois optimale en nombre de tour, dont le nombre d'éléments échangés ne dépend pas de la taille des messages et dont la sécurité est prouvée dans le modèle standard. Nous obtenons ce schéma de signature en suivant un framework établi par

Fischlin dans [Fis06] que nous adaptons afin d'obtenir une primitive intermédiaire : la signature sur chiffré randomisable. Un tel schéma permet de signer une classe de messages correspondant aux chiffrés d'un même message. Cela permet alors, à partir d'une signature sur un message chiffré, d'obtenir une signature sur n'importe quelle randomisation du même message chiffré. Ainsi, en prouvant la possession d'une telle signature randomisée à l'aide d'un schéma de preuve à divulgation nulle de connaissance non intéractif (NIZK), nous pouvons garantir la propriété de *Blindness*.

— *CROOT : Code-based Round-Optimal Oblivious Transfer* avec Nicolas Aragon, Olivier Blazy et Philippe Gaborit, publié.

Cette publication nous a permis de présenter un protocole d'*Oblivious Transfer* à base de cryptographie basée sur les codes correcteurs d'erreurs tout en étant optimal en nombre de tours. Dans notre protocole, deux utilisateurs, l'envoyeur, possédant deux messages m_0, m_1, et le receveur, possédant un bit b, interagissent. À la fin de l'exécution, le receveur récupère le message m_b tandis que l'envoyeur n'apprend aucune information sur le bit b.

Nous avons construit notre schéma à partir d'un framework nécessitant du chiffrement asymétrique possédant une propriété particulière sur les clés publiques utilisées et des fonctions de hachage. Si ce framework utilisé pour notre travail était déjà référencé dans la littérature, nous proposons une adaptation de celui-ci à l'aide d'une nouvelle fonctionnalité idéale pour l'*Oblivious Transfer*. Ce faisant, la sécurité du schéma est prouvée dans le modèle de la composabilité universelle (*Universal Composability*, introduit dans [Can01]) tout en étant basée sur des hypothèses de difficulté résistantes à l'ordinateur quantique.

Finalement, nous proposons des évaluations de performances de nos différents algorithmes d'*Oblivious Transfer*, implémentés en langage C, montrons une efficacité tout à fait pratique.

— *Secure Kafka Zookeeper Architecture* avec Olivier Blazy, Emmanuel Conchon, Mathieu Klingler et Damien Sauveron, en cours de soumission.

Un peu différent des autres travaux présentés ici, nous avons cherché à appliquer certains outils cryptographiques intéressants à une architecture de *Publish/Subscribe* largement utilisée dans l'industrie, Apache Kafka, afin de la rendre plus sécurisée. Une telle architecture permet de mettre en relation des producteurs de contenu (*publishers*) et des consommateurs (*subscribers*). Notamment, nous avons plusieurs buts précis à atteindre ; le premier concerne la confidentialité des publications (*publication confidentiality*) et des sujets de publications (*topic confidentiality*). En effet, il est primordial de faire en sorte que ni les publications ni les sujets les concernant ne doivent être accessibles à quelqu'un qui n'est pas autorisé par une politique d'accès particulière. Pour ce faire, nous agrémentons Apache Kafka d'une utilisation de chiffrement par attributs (*Attribute-Based Encryption*) qui ne permet qu'à certains utilisateurs, possédant une clé associée à des attributs, de déchiffrer des clés secrètes de chiffrement AES. En outre, nous garantissons également la *subscription privacy*, le fait de pouvoir s'abonner à un sujet sans le divulguer à

personne, à l'aide d'un algorithme de récupération privée de l'information (*Private Information Retrieval*).

Pour ce travail, nous avons également proposé des mesures de performance en branchant à Apache Kafka et Apache Zookeeper les algorithmes de cryptographie nécessaires. Notre construction octroie de très bonnes performances d'utilisation si l'on restreint le nombre d'attributs dans l'utilisation du chiffrement et déchiffrement *ABE*, nombre qui correspond tout à fait à des scénarios pratiques.

Nous ne présenterons pas ce travail par la suite car il n'est pas suffisamment proche des autres dans son approche et dans les techniques utilisées.

1.4 Organisation

La suite de la thèse comprend cinq autres chapitres. Le chapitre 2 introduit des éléments techniques de base nécessaires en cryptographie : des éléments de notation mais également comment modéliser la menace adversariale, sur quoi repose la sécurité et comment la prouver. Le chapitre 3 définit les différentes primitives et outils cryptographiques qui seront utile dans nos constructions puis donne des instantiations classiques. Les trois suivants (chapitres 4, 5 et 6) correspondent chacun à un des travaux présentés dans la partie précédente concernant les contributions.

Un point important à soulever est celui de la ou plutôt des langues utilisées pour la rédaction. Si le français est le candidat naturel, la cryptographie moderne se fait en anglais et bon nombre de termes ne trouvent pas une traduction facile et compréhensible. Par conséquent, pour ne pas être confronté à une lecture tout simplement fastidieuse des prochains chapitres du fait d'un langage mi-français mi-anglais, faussement cohérent et parfaitement odieux, l'anglais technique et international est conservé.

Chapter 2

Technical Outline

Before delving into the design of our cryptographic schemes, we have to brush the foundations on which they are built. In order to do this, we introduce mathematical and algorithmical notations used troughout the rest of this thesis. After, we introduce core notions when dealing with security: how we model adversaries, what are their limits and on which assumptions can we claim security. Foundations of cryptography could be many books in itself but we present briefly what we used in our works such as computationial hardness, security games, the Random Oracle model and the Universal Composability framework.

Contents

2.1 Notations

In our work, we tried to follow notation conventions used by other researchers when dealing with number theoretic settings.

— Groups will be noted by bold letters such as $\mathbb{G}$ or $\mathbb{Z}$. When pairings are considered, the groups will be $\mathbb{G}_1$ and $\mathbb{G}_2$ with the target group $\mathbb{G}_T$.

— Group generators will be noted by lowercase letters ($g \in \mathbb{G}$). If needed to indicate their group, an indice can be added (e.g. g_1, g_2).

Matricial Notations. If $\mathbf{A} \in \mathbb{Z}_p^{(k+1)\times n}$ is a matrix, then $\mathbf{A} \in \mathbb{Z}_p^{k\times n}$ denotes the upper matrix of $\mathbf{A}$ and $\underline{\mathbf{A}} \in \mathbb{Z}_p^{1\times n}$ denotes the last row of $\mathbf{A}$. We use classical notations from [GS08] for operations on vectors (. for the dot product and $\odot$ for the product

component-wise). Concatenation of matrices having the same number of lines will be denoted by $\mathbf{A}||\mathbf{B}$ (where $a||b+c$ should be implicitly parsed as $a||(b+c)$).

We use implicit representation of group elements as introduced in [EHK+13]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$ define $[a]_s = g_s^a \in \mathbb{G}_s$ as the *implicit representation* of a in $\mathbb{G}_s$ (we use $[a] = g^a \in \mathbb{G}$ if we consider a unique group). More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{n \times m}$ we define $[\mathbf{A}]_s$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}_s$:

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

We will always use this implicit notation of elements in $\mathbb{G}_s$, i.e., we let $[a]_s \in \mathbb{G}_s$ be an element in $\mathbb{G}_s$. Note that from $[a]_s \in \mathbb{G}_s$ it is generally hard to compute the value a (discrete logarithm problem in $\mathbb{G}_s$). Further, from $[b]_T \in \mathbb{G}_T$ it is hard to compute the value $[b]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$ (pairing inversion problem). Obviously, given $[a]_s \in \mathbb{G}_s$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[ax]_s \in \mathbb{G}_s$. Further, given $[a]_1, [b]_2$ one can efficiently compute $[ab]_T$ using the pairing e. For $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^k$ define $e([\mathbf{a}]_1, [\mathbf{b}]_2) := [\mathbf{a}^\top \mathbf{b}]_T \in \mathbb{G}_T$.

2.2 Foundations

As security is at the core of Cryptography, it is extremely important to present the assumptions we make and how we evaluate the many tools used and created. What is permitted to adversaries and users? What are their limits in terms of computation, memory, bandwidth and interactions? Another concern is to make asymmetric cryptographic schemes' security rely under very hard to solve mathematical problems. The intuition behind is that schemes should be as hard to break as the problems themselves. Immense work has been done by the Cryptographic community to unify as much as possible, to give a common ground for security evaluation and we will try to give at least what is needed to contextualize our work.

Ideally, an adversary attacking an encrypted message should not learn any bit of information about it regardless of his computational power: this is, informally, information-theoretic security. Attaining such level of security is far too difficult therefore there is a need to model a powerful yet limited adversary. Thus, reducing a successful adversarial attack to solving a difficult mathematical problem becomes an important tool to prove the security of a cryptographic scheme. From [KL14], this means two relaxations are necessary regarding ideal information-theoretic security:

— Security is only guaranteed against adversaries running in a practicable amount of time. With unlimited computation time schemes can all be broken. However, setting unrealistic amount of resources required to violate them is a solution.

— There is a probability an attack can succeed. The goal is to have this probability as insignificant as possible.

2.2.1 Computational Difficulty

Number theory offers particular settings in which hard problems exist. Furthermore, these settings not only provide security reductions but also special mathematical operations allowing many useful cryptographic features (confidentiality, non-repudiation, integrity, ...). We present here all the classical "hardness" assumptions used in our designs plus their matrix variants.

Classical Hardness Assumptions

Definition 1 (Decisional Diffie-Hellman (DDH))**.** *Let $\mathbb{G}$ be a cyclic group of prime order p. The* DDH *assumption states that given $(g, g^a, g^b, g^c) \in \mathbb{G}$, it is hard to determine whether $c = ab$.*

Definition 2 (External Diffie-Hellman (XDH [BBS04]))**.** *This variant of* DDH*, states that while the* DDH *is easy in $\mathbb{G}_2$,* DDH *is hard in $\mathbb{G}_1$.*

Definition 3 (Symmetric External Diffie-Hellman (SXDH [ACHdM05]))**.** *This variant of* DDH*, used mostly in bilinear groups in which no computationally efficient homomorphism exists from $\mathbb{G}_2$ to $\mathbb{G}_1$ or $\mathbb{G}_1$ to $\mathbb{G}_2$, states that* DDH *is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

We recall the definition of the matrix Diffie-Hellman (MDDH) assumption [EHK+13].

Definition 4 (Matrix Distribution)**.** *Let $k \in \mathbb{N}$. We call $\mathcal{D}_k$ a matrix distribution if it outputs matrices in $\mathbb{Z}_p^{(k+1)\times k}$ of full rank k in polynomial time.*

Without loss of generality, we assume the first k rows of $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$ form an invertible matrix. The $\mathcal{D}_k$-Matrix Diffie-Hellman problem is to distinguish the two distributions $([\mathbf{A}], [\mathbf{Aw}])$ and $([\mathbf{A}], [\mathbf{u}])$ where $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

Definition 5 ($\mathcal{D}_k$-Matrix Diffie-Hellman Assumption $\mathcal{D}_k$-MDDH)**.** *Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2, T\}$. We say that the $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-*MDDH*) Assumption holds relative to* GGen *in group $\mathbb{G}_s$ if for all PPT adversaries $\mathcal{D}$,*

$$\begin{aligned}\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) &\stackrel{\mathsf{def}}{=} |\Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{Aw}]_s) = 1] \\ &\quad - \Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{u}]_s) = 1]| \\ &= \mathsf{negl}(\lambda),\end{aligned}$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k, \mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k, \mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

Definition 6 ($\mathcal{D}_k$-Kernel Diffie-Hellman Assumption $\mathcal{D}_k -$ KerMDDH)**.** *Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2\}$. We define $\mathsf{Adv}^{kmddh}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D})$ by*

$$\Pr[\mathbf{c}^\top \mathbf{A} = \mathbf{0} \wedge \mathbf{c} \neq \mathbf{0} | [\mathbf{c}]_{3-s} \xleftarrow{\$} \mathcal{D}(\mathcal{G}, [\mathbf{A}]_s)]$$

*where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$. We say that the $\mathcal{D}_k$-Kernel Diffie-Hellman Assumption ($\mathcal{D}_k -$*KerMDDH*) assumption holds relative to* GGen *in group $\mathbb{G}_s$, if for all PPT adversaries $\mathcal{D}$,*

$$\mathsf{Adv}^{kmddh}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) = \mathsf{negl}(\lambda)$$

In the following, we write $k -$ MDDH for $\mathcal{D}_k -$ MDDH. It should be noted that $1 -$ MDDH is the SXDH assumption used in this paper.

Quantum Resistant Assumptions

Many hard problems are known to be resistant to Shor's quantum algorithm. Here, we present two of them related to the field of Cryptography based on Error-Correcting Codes.

Definition 7 (Syndrome Decoding (**SD**))**.** *Given a full-rank matrix* $\boldsymbol{H} \in \mathbb{F}^{(n-k)\times n}$*, a syndrome* $\boldsymbol{s} \in \mathbb{F}^{n-k}$ *and a weight* w*, it is hard to find a vector* $\boldsymbol{x} \in \mathbb{F}^n$ *of weight lower than* w *such that* $\boldsymbol{H}\boldsymbol{x}^\top \leqslant \boldsymbol{s}^T$*.*

Definition 8 (Decisional **SD** Problem)**.** *On input* $(\mathbf{H}, \mathbf{s}^\top) \in \mathbb{F}^{(n-k)\times n} \times \mathbb{F}^{(n-k)}$*, the* Decisional SD Problem $\mathsf{DSD}(n, k, w)$ *asks to decide with non-negligible advantage whether* $(\mathbf{H}, \mathbf{s}^\top)$ *came from the* $\mathsf{SD}(n, k, w)$ *distribution or the uniform distribution over* $\mathbb{F}^{(n-k)\times n} \times \mathbb{F}^{(n-k)}$*.*

These problems can be specialized for both the Hamming metric and the rank metric: in the Hamming metric, $\mathbb{F} = \mathbb{F}$ while in the rank metric, $\mathbb{F} = \mathbb{F}_{q^m}$.

2.2.2 Random Oracle vs Standard Model

Historically, cryptographers began to prove in an idealized model called the *Random Oracle Model* (ROM) where there exists an ideal function H that when queried on x returns a truly random output $H(x)$. Due to the random oracle properties (*uniformity*, *programmability* and *extractability*) it allows very efficient and simple schemes to be proven secure. The issure here is that this ideal function H is not known to be practically instantiable even using cryptographically secure hash functions that may mimic them in some ways. Therefore, ROM proofs does not offer immediate practical security but rather a belief that the design of the scheme is on the "right" way.

A more real-world approach to proofs is found in the Standard Model (SM). In it, the prover has also the need to reduce breaking the security of the scheme to solving a hard problem but no random oracles can be used in the proof. That makes the proof much harder to obtain and can be seen as a **very desirable goal** for the protocol designer.

2.2.3 Security Games

Security games or experiments are a way to express what an adversary can do, how the challenges are presented to him and how he succeeds rather than defining a specific attack against a scheme. Security is then evaluated through the advantage, the probability of winning, the adversary has. This advantage is often expressed with regards to the probability of solving a hard problem and for security to hold, it should be negligible. Formally, we give the definition of a **negligible function** in definition 9 and such a function can be noted $\mathsf{negl}(n)$.

Definition 9 (Negligible function)**.** *A function f is* **negligible** *if for every polynomial p there is an N such that for all* $n > N$ *it holds that* $f(n) < \frac{1}{p(n)}$*.*

$\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(M_0, M_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{pk})$
4. $c^* \leftarrow \mathsf{Enc}(\mathsf{ek}, M_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*)$
6. `RETURN` b'

Figure 2.1 – $\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathcal{E},\mathcal{A}}$ experiment.

A straightforward example would be the *Indistinguishability* under Chosen-Plaintext Attack for an encryption scheme. In this experiment, a key k is chosen uniformly at random then the adversary has access to a "black-box" encryption oracle under the key k. That is to say, he can use this oracle to encrypt any messages he wants until he outputs two messages m_0, m_1. After that, a secret bit b is randomly selected and the adversary is given $\mathsf{Enc}_k(m_b)$. To win the game, he has to output a bit $b' = b$, otherwise he loses.

Definition 10 (IND-CPA security)**.** *An encryption scheme $\mathcal{E}$ is Indistinguishable under chosen-plaintext attack or* IND-CPA*-secure if for all probabilistic polynomial time adversaries $\mathcal{A}$ there is a negligible function* negl *such that*

$$\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathcal{E},\mathcal{A}}(n) = 1] <= \frac{1}{2} + \mathsf{negl}(n)$$

where the probability is taken over the randomness of $\mathcal{A}$ and the randomness used in the experiment.

The IND-CPA experiment is formally defined in figure 2.1. While this is the nowadays a minimal requirement in terms of security for an encryption scheme, it shows us that we can adapt this type of experiment to further definitions of security or other kinds of schemes. We present in later sections experiments associated to the cryptographic primitives we use and design.

2.2.4 Universal Composability

The Universal-Composability (UC) framework was introduced in [Can01].

In the context of multi-party computation, one wants several users P_i with inputs x_i to be able to compute a specific function $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ without learning anything except y_i. This approach was seen for example in Yao's Millionaires' problem [Yao82b], where two millionaires want to know who is richer without revealing their respective wealth. So here, x_i is the wealth of the millionaire i, and f simply returns which one is richer (in this specific case $y_1 = y_i = y_n$).

Instead of following the classical approach which aims to list exhaustively all the expected properties, Canetti did something else and tried to define how a protocol should ideally work.

For that, he divided the world into two spaces, the real world, where the protocol is run with some possible attack, and the ideal world where everything would go smoothly. For a good protocol, it should be impossible to distinguish the real world from the ideal one.

In the ideal world there is an incorruptible entity named the ideal functionality, to which players can send their inputs privately, and then receive the corresponding output without any kind of communication between the players. This way the functionality can be set to be correct, without revealing anything except what is expected.

A protocol, in the real world with an adversary, should create an execution similar to the one obtained by the ideal functionality. This means that the communication between the players should not give more information than the functionality description, and its output. In this case the protocol runs not really against the adversary but against the environment who picks the inputs given to the players, and obtains the outputs. After the interaction the environment should output a bit saying whether he is in the real world, as shown in figure 2.2.

The main constraint is that the adversary is now free to interact with the environment whenever he wants which prevents the simulator from rewinding when needed. The adversary has access to the communication between the players but not their inputs/outputs, while the environment has only access to the inputs/outputs.

To prove that a protocol realizes the ideal functionality, we consider an environment $\mathcal{Z}$ which can choose inputs given to all the users and whose goal is to distinguish in which case he receives outputs from the real world execution with an adversary $\mathcal{A}$, and in which case they come from an ideal execution with an ideal adversary $\mathcal{S}$ who interacts solely with the functionality. Such protocol realizes the functionality if for all polynomial adversary $\mathcal{A}$, there exists a polynomial simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish the real world from the ideal one with a non-negligible probability.

In the adaptive corruption setting, the adversary can get complete access to the private credentials and the internal memory of an honest player, and then get control of it, at any time.

More precisely, we will work in the UC framework with joint state proposed by Canetti and Rabin [CR03] (for the CRS). Informally, this allows different protocols to have some common states while preserving their security. Basically for a given session identifier sid we also define sub-session identifier ssid, and so we have a functionality, possibly generated on the fly, for each ssid.

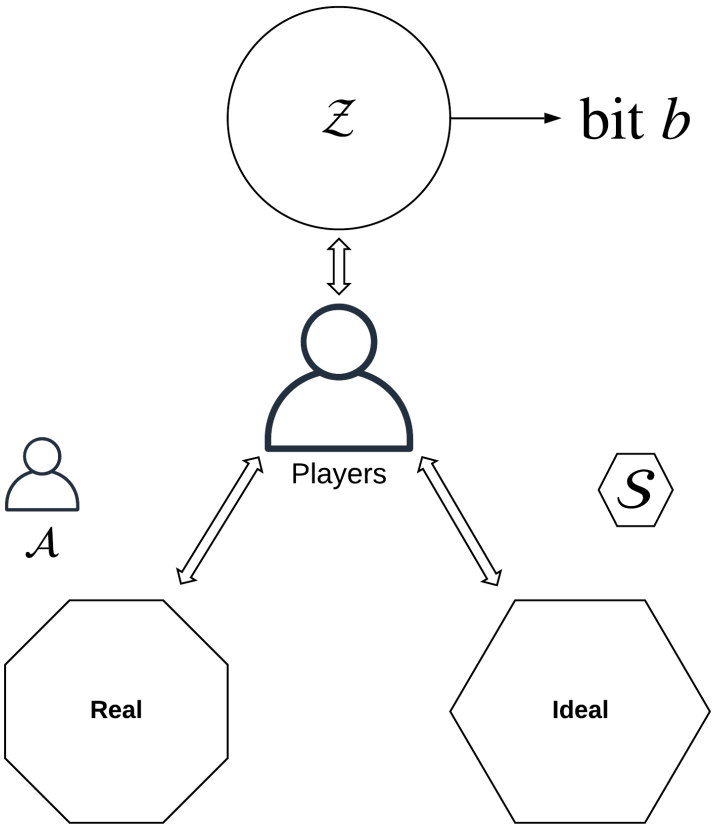

Figure 2.2 – UC model

Chapter 3

Cryptographic Tools

Designing cryptographic primitives does not necessarily mean working from a blank slate. Many advanced tools may be construced by combining and tweaking existing schemes with particular properties. Thus, we define here many tools that will be used troughout the rest of this thesis as modular building blocks before giving a few classical instantiations. Thus, definitions of asymmetric encryption, digital signature, Non-Interactive Zero Knowledge proof system and Smooth Projective Hash Function are given.

The main encryptions schemes used are El Gamal and Cramer-Shoup due to their malleability property and sufficient security requirements. As for digital signatures, we needed the BLS signature scheme for its simple pairing-based setting but the Structure-Preserving signature scheme of [KPW15] will be our main tool.

Contents

3.1 Encryption

From [BFPV11], an encryption scheme $\mathcal{E}$ is described through four algorithms (Setup, KeyGen, Enc, $\mathsf{Decrypt}$):

$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(M_0, M_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{ek}, \mathsf{ODecrypt}(\cdot))$
4. $c^* \leftarrow \mathsf{Enc}(\mathsf{ek}, M_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot))$
6. `IF` $(c^*) \in \mathcal{CT}$ `RETURN` 0
7. `ELSE RETURN` b'

Figure 3.1 – $\mathsf{Exp}^{\mathsf{IND\text{-}CCA}-2}_{\mathcal{E},\mathcal{A}}$ experiment.

— $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;

— $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of keys, a (public) encryption key ek and a (private) decryption key dk;

— $\mathsf{Enc}(\mathsf{ek}, M; \rho)$ outputs a ciphertext $c = \mathcal{C}(M)$, on the message M, under the encryption key ek, with the randomness ρ;

— $\mathsf{Decrypt}(\mathsf{dk}, c)$ outputs the plaintext M, encrypted in the ciphertext $\vec{c}$ or $\perp$.

In addition to correctness, we will expect from encryption schemes the $\mathsf{IND\text{-}CCA} - 2$ property (Indistinguishability under Adaptive Chosen-Ciphertext Attack, introduced in [RS92]):

— *Correctness*: For every pair of keys $(\mathsf{ek}, \mathsf{dk})$ generated by KeyGen, every messages M, and every random ρ, we should have $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, M; \rho)) = M$.

— $\mathsf{IND\text{-}CCA} - 2$: This notion states that an adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two original plaintexts, and can ask several decryption of ciphertexts as long as they are not the challenge one.

In this security game, defined in figure 3.1, the $\mathsf{ODecrypt}$ oracle outputs the decryption of c under the challenge decryption key dk. The input queries (c) are added to the list $\mathcal{CT}$ of decrypted ciphertexts.

3.2 Digital Signature

A digital signature scheme ($\mathcal{S}$ [DH76,GMR88]) allows a signer to produce a verifiable proof that he indeed produced a message. It is described through four algorithms:

— $\mathsf{Setup}(1^{\mathfrak{K}})$ where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme, for example the message space;

— $\mathsf{KeyGen}(\mathsf{param})$, outputs a pair of $(\mathsf{sk}, \mathsf{vk})$, where sk is the (secret) signing key, and vk is the (public) verification key;

— $\mathsf{Sign}(\mathsf{sk}, M; \mu)$, outputs a signature σ, on a message M, under the signing key sk, and some randomness μ;

— $\mathsf{Verify}(\mathsf{vk}, M, \sigma)$ checks the validity of the signature σ with respect to the message M and the verification key vk. And so outputs a bit.

From signatures, we expect at least the following properties:

— *Correctness*: For every pair $(\mathsf{vk}, \mathsf{sk})$ generated by KeyGen, for every message M, and for all randomness μ, we have $\mathsf{Verify}(\mathsf{vk}, M, \mathsf{Sign}(\mathsf{sk}, M; \mu)) = 1$.

— *Existential Unforgeability under Chosen-Message Attacks [GMR88]* ($\mathsf{EUF} - \mathsf{CMA}$). Even after querying n valid signatures on chosen messages (M_i), an adversary should not be able to output a valid signature on a fresh message M.

3.3 Zero-Knowledge

3.3.1 Groth-Sahai Commitments

In [GS08], Groth and Sahai proposed non-interactive zero-knowledge proof systems for bilinear groups. It allows a prover to convince a verifier that he possesses group elements or scalars satisfying equations of a particular form. For our work, we mainly use the satisfiability of pairing product equations. Various instantiations were proposed in this seminal paper based on common hardness assumptions such as DLin and SXDH.

Initialization We work in a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. The commitment keys for group $\mathbb{G}_1$ is $\mathbf{u} = (\mathbf{u_1}, \mathbf{u_2})$. We initialize it with random values $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$ as $\mathbf{u_1} = [1, \alpha]_1, \mathbf{u_2} = [\beta, \alpha\beta]_1$. Hence, $\mathbf{u}$ is a Diffie-Hellman tuple in $\mathbb{G}_1$. This commitment is binding but can be set to hiding if we define instead $\mathbf{u_2} = \beta\mathbf{u_1} - (1, g_1)$. The commitment key $\mathbf{v}$ can be analogously defined for $\mathbb{G}_2$.

Group element commitment. To commit to a group element $\mathcal{X} \in \mathbb{G}_1$, using randomness $r_1, r_2 \in \mathbb{Z}_p$,

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

Scalar commitment. To commit to a scalar $x \in \mathbb{Z}_p$, using randomness $r \in \mathbb{Z}_p$,

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

Proofs. Under the SXDH assumption, the two initializations of the commitment key (perfectly binding or perfectly hiding) are indistinguishable. A Groth-Sahai proof is a pair of elements $(\pi, \theta) \in \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$. These elements are constructed to help verifying pairing relations on committed values. Being able to produce a valid pair implies knowing plaintexts verifying the appropriate relations. We will note $\langle \mathcal{X} \rangle_1$ for a committed group element in $\mathbb{G}_1$ and $\langle x \rangle_2$ for a committed scalar x in $\mathbb{G}_2$.

Throughout the paper, we are going to encounter two kinds of relations.

— Linear Pairing Product Equations in $\mathbb{G}_1$: $[\sum_i X_i \mathcal{B}_\mathbf{i}]_T = [t]_T$ to prove that the committed group elements $[X_i]_1$ satisfy a pairing product relation with constants

vector $[\mathcal{B}_\mathbf{i}]_2$ equal to a target group element in $\mathbb{G}_T$. In this particular case, the proof is composed only of $\theta \in \mathbb{G}_2^2$.

— Multi-scalar multiplications equations: $[y\mathcal{A}]_1 = [\mathcal{X}]_1$ to prove that the committed scalar in $\mathbb{G}_2$ is the discrete log of the committed element $\mathcal{X}$ committed in $\mathbb{G}_1$.

3.4 Hash Functions

A hash function H is a function mapping an input from $\{0,1\}^*$ to a fixed-length input $\{0,1\}^l$ where l is the dependent on the security parameter chosen. The output, of size l, is called the *digest*. The main property we need from a hash function is **collision resistance**.

Definition 11 (Collision-Resistant)**.** *A hash function is said to be collision-resistant if for any adversary $\mathcal{A}$, it is hard to find a collision. More precisely, we denote $\mathsf{Succ}_H^{coll}(\mathcal{A}) = \Pr[(m_0, m_1) \leftarrow \mathcal{A}(H) : H(m_0) = H(m_1)]$ the success probability of an adversary finding two inputs generating the same output. Such probability should be negligible.*

3.5 Smooth Projective Hash Function

SPHF systems have been defined by Cramer and Shoup [CS02] in order to build a chosen-ciphertext secure encryption scheme. They have thereafter been extended and applied to several other primitives. Such a system is defined on a language L, with five algorithms:

— $\mathsf{Setup}(1^\mathfrak{K})$ generates the system parameters, according to a security parameter $\mathfrak{K}$;

— $\mathsf{HashKG}(L)$ generates a hashing key hk for the language L (depending on the earlier param);

— $\mathsf{ProjKG}(\mathsf{hk}, L, W)$ derives the projection key hp, using hk and possibly depending on a word W;

— $\mathsf{Hash}(\mathsf{hk}, L, W)$ outputs the hash value from the hashing key;

— $\mathsf{ProjHash}(\mathsf{hp}, L, W, w)$ outputs the hash value from the projection key and the witness w such that $W \in L$.

The correctness of the scheme assures that if W is in L with w as a witness, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, L, W) = \mathsf{ProjHash}(\mathsf{hp}, L, W, w)$. In our setting, these hash values will belong to a group $\mathbb{G}$. The security is defined via the *smoothness* property. It guarantees that if $W \notin L$, the hash value is *statistically* indistinguishable from a random element, even knowing hp.

Following a classification detailed in [BBC+13b], we are going to work solely with a subclass of SPHF (called *KV-SPHF*), where the ProjKG algorithm no longer takes the word W as a parameter. This allows for concurrent rounds in a protocol, since there is no need to wait for the first message (the encryption) to be delivered before sending the second one (the projected key) since the computation of this key does not depend on the first message.

3.6 Instantiations

In this section, we present instantiations of digital signature and encryption schemes we used in later chapters.

3.6.1 BLS Signatures

[BLS04] This digital signature scheme makes use of bilinear pairings. It produces short signatures and is provably secure ($\mathsf{EUF}-\mathsf{CMA}$ security) assuming both the existence of random oracles and the intractability of the CDH. Signatures being only one element of a group $\mathbb{G}_1$ and the verification algorithm necessitating one pairing computation make the scheme really efficient four our Secret Handshake protocol.

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a non-degenerate, efficiently computable, bilinear pairing where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order p. Let $\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$ be a hash function. The scheme can be described through three algorithms:

— $\mathsf{Setup}(1^{\mathfrak{K}})$ generates the verification key $\mathsf{vk} = [x]_2$ where x is a random element of $\mathbb{Z}_p$ and $\mathsf{sk} = x$ is the signing key;

— $\mathsf{Sign}(\mathsf{sk}, m)$ generates a signature $\sigma = [\mathcal{H}(m)x]_1$;

— $\mathsf{Verify}(\mathsf{vk}, \sigma, m)$ accepts if $[\sigma]_T = [\mathcal{H}(m)\mathsf{vk}]_T$;

3.6.2 Structure-Preserving Signatures

We recall here the structure-preserving signature of [KPW15] in Figure 3.2. As our use case is concrete, we fix their two parameters: both n and k equal 1. For some of our applications, though, we do not necessarily need a full fledged structure-preserving signature but just handling signatures of group elements. For these reasons we can give our τ as a scalar, removing the need for σ_3 and σ_4:

$$\sigma := ([(1, \mathbf{m})\mathbf{K} + \mathbf{r}^\intercal(\mathbf{P_0} + \tau\mathbf{P_1})]_1, [\mathbf{r}^\intercal\mathbf{B}^\intercal]_1, \tau)$$

The verification is then simplified to the single equation:

$$[\sigma_1 \cdot \mathbf{A}]_T = [(1, \mathbf{m}) \cdot \mathbf{C} + \sigma_2 \cdot (\mathbf{C_0} + \tau\mathbf{C_1})]_T$$

3.6.3 El Gamal Encryption

The four algorithms of this encryption scheme [ElG84] are described as follows:

— $\mathsf{Setup}_{\mathcal{E}}(1^{\mathfrak{K}})$: outputs $\mathsf{param} = (\mathbb{G}, p, [1])$.

— $\mathsf{KeyGen}_{\mathcal{E}}(\mathsf{param})$: outputs $(\mathsf{ek}, \mathsf{dk}) = ([h], h)$ with $h \xleftarrow{\$} \mathbb{Z}_p$.

— $\mathsf{Enc}(\mathsf{ek}, [m]; r)$ outputs $c = (c_1, c_2) = ([r, rh + m])$ with $r \xleftarrow{\$} \mathbb{Z}_p$.

— $\mathsf{Decrypt}(\mathsf{dk}, c)$ computes $[c_2 - hc_1] = [m]$, outputs $[m]$.

This scheme is semantically secure against chosen-plaintext attacks ($\mathsf{IND\text{-}CPA}$) under DDH.

KeyGen:
$\mathbf{A}, \mathbf{B} \xleftarrow{\$} \mathcal{D}_1; \mathbf{K} \xleftarrow{\$} \mathbb{Z}_q^{2\times 2}$
$\mathbf{K_0}, \mathbf{K_1} \xleftarrow{\$} \mathbb{Z}_q^{2\times 2}$
$\mathbf{C} \stackrel{\text{def}}{=} \mathbf{KA} \in \mathbb{Z}_q^{2\times 1}$
$(\mathbf{C_0}, \mathbf{C_1}) \stackrel{\text{def}}{=} (\mathbf{K_0 A}, \mathbf{K_1 A}) \in (\mathbb{Z}_q^{2\times 1})^2$
$(\mathbf{P_0}, \mathbf{P_1}) \stackrel{\text{def}}{=} (\mathbf{B}^\intercal \mathbf{K_0}, \mathbf{B}^\intercal \mathbf{K_1}) \in (\mathbb{Z}_q^{1\times 2})^2$
$\mathbf{sk} \stackrel{\text{def}}{=} (\mathbf{K}, [\mathbf{P_0}]_1, [\mathbf{P_1}]_1, [\mathbf{B}]_1)$
$\mathbf{vk} \stackrel{\text{def}}{=} ([\mathbf{C_0}]_2, [\mathbf{C_1}]_2, [\mathbf{C}]_2, [\mathbf{A}]_2)$

Sign:
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q; \tau \xleftarrow{\$} \mathbb{Z}_q$
$\sigma_1 \stackrel{\text{def}}{=} [(1, \mathbf{m})\mathbf{K} + \mathbf{r}^\intercal(\mathbf{P_0} + \tau \mathbf{P_1})]_1 \in \mathbb{G}_1^{1\times 2}$
$\sigma_2 \stackrel{\text{def}}{=} [\mathbf{r}^\intercal \mathbf{B}^\intercal]_1 \in \mathbb{G}_1^{1\times 2}$, $\sigma_3 \stackrel{\text{def}}{=} [\mathbf{r}^\intercal \mathbf{B}^\intercal \tau]_1 \in \mathbb{G}_1^{1\times 2}$
$\sigma_4 \stackrel{\text{def}}{=} [\tau]_2 \in \mathbb{G}_2$
$\texttt{Return}(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$

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

Figure 3.2 – The structure-preserving signature algorithms from [KPW15]

3.6.4 Cramer Shoup

The Cramer-Shoup (CS), [CS98], encryption scheme can be turned into a labeled public-key encryption scheme:

— $\mathsf{Setup}(1^\mathfrak{K})$ generates a group $\mathbb{G}_1$ of order p.

— $\mathsf{KeyGen}(\mathsf{param})$ generates $(g_1, g_2) \xleftarrow{\$} \mathbb{G}_1^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = [x_1 g_1 + x_2 g_2]_1$, $d = [y_1 g_1 + y_2 g_2]_1$, and $h = [z g_1]_1$. It also chooses a collision resistant hash function H_k in a family of hash functions $\mathfrak{H}_K$. The encryption key is $\mathsf{ek} = ([g_1, g_2, c, d, h]_1, H_k)$.

— $\mathsf{Enc}(\ell, \mathsf{ek}, M; r)$, for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, u = [r \cdot g_1, r \cdot g_2]_1, e = [M + rh]_1, v = [r(c + \chi r)]_1)$, where v is computed afterwards with $\chi = H_k(\ell, u, e)$.

— $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$, one first computes $\chi = H_k(l, u, e)$ and checks whether $[(x_1 + \chi y_1)u_1 + (x_2 + \chi y_2)u_2]_1 \stackrel{?}{=} [v]_1$. If the equality holds, one computes $M = [e - z u_1]_1$ and outputs M. Otherwise, one outputs $\perp$.

This scheme is indistinguishable against chosen-ciphertext attacks ($\mathsf{IND\text{-}CCA}$), under the DDH assumption and if one uses a collision-resistant hash function H_k.

Chapter 4

Round-Optimal Constant-Size Blind Signatures

In this chapter, we give an overview of Blind Signature (BS) schemes before presenting various tools that allow us to conceive our own scheme with desirable properties, leading to a publication in the 17th International Conference on Security and Cryptography (SECRYPT 2020).

Blind signature schemes introduced in [Cha82] are a variant of digital signature schemes, with an additional property, called the *blindness* which transforms the classical non-interactive signature scheme in an interaction between a user, providing a message, and a signer, providing a signature. In this protocol, the signer should not learn any information concerning the messages he signs, he is "blind". More precisely, the view of the signer should be unlinkable to the (message/signature) pairs resulting from several executions of the protocol.

They were introduced as a fundamental building block for applications that guarantee user anonymity, e.g. e-cash [Cha82, CFN90, OO92, CHL05, FPV09], e-voting solutions [FOO93, BFPV11, BPV12a], and anonymous credentials [Bra94, CL01, BCC$^+$09, Fuc11]. Because they are of practical interest, it is highly important that such schemes offer high efficiency communication-wise ; our construction achieves such efficiency both in terms of number of rounds and amount of data exchanged.

Contents

4.1 General overview

4.1.1 State of the art

The first constructions for blind signature schemes (such as [Oka93, Oka06]) were greatly interactive, until Fischlin proposed a generic framework for round-optimal blind signature schemes in [Fis06], in the sense that only two flows of communication between the user and the signer are needed. Many efficient instantiations of this framework have been proposed although at the cost of exchanging information of size depending on the message length [BFPV11, BPV12b, BPV12a], or by relying on the Random Oracle Model [PS00, Abe01, BL13, HKL19].

To the best of our knowledge, constant-size instantiations in the standard model (without any random oracle) were given in [AFG+10, Gha17]. The former is given in a pairing-based setting with a final signature consisting of 18 elements in the first group $\mathbb{G}_1$ and 16 in the second one $\mathbb{G}_2$, but relies on a new ad-hoc q-type assumption. In [Gha17], the proposed blind signature is in the standard model but under a non standard q-type assumption: The *Blind Signature One More*, which basically assumes the security of the scheme and could likely only be proven in the generic model. In [BBCF20], autors proposed a new blind signature scheme, which is round-optimal, constant-size, in the standard model and with a classical assumption.

4.1.2 Definitions and Security

Resulting from the definition of a digital signature scheme, given in section 3.2, we have for the Blind Signature the following formal definition.

Definition 12 (Blind signature scheme)**.** *A blind signature scheme is defined by three polynomial time algorithms and one interactive polynomial time protocol* $\mathcal{BS} = (\mathsf{BSSetup}, \mathsf{BSKeyGen}, \mathsf{BSProtocol}\langle\mathcal{S},\mathcal{U}\rangle, \mathsf{Verify})$.

— $\mathsf{BSSetup}(\mathfrak{K})$ *outputs the parameters* param *of the scheme.*

— $\mathsf{BSKeyGen}(\mathsf{param})$ *outputs a pair of keys, the (private) signature key* sk *and the (public) verification key* vk.

— $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}),\mathcal{U}(\mathsf{vk},m)\rangle$ *is an interactive protocol between user* $\mathcal{U}$ *and signer/server* $\mathcal{S}$*. It produces a signature* σ *on* m *valid under* vk.

— $\mathsf{Verify}(\mathsf{vk},m,\sigma)$ *checks if* σ *is a valid signature with regards to* vk*. It outputs 1 if* σ *is valid, 0 otherwise.*

The second security property for blind signatures is a notion of *unforgeability*, which intuitively means that after n interactions, a user should not obtain more than n signatures on different messages. This property has been formalized in [PS00], motivated by the use of blind signatures for e-cash : a user should not be able to produce more (message/signature) pairs (coins) than the number of signing executions with the bank (withdrawals). The security model was further revisited in [SU12] for other contexts.

As said in chapter 2, for security reasons, it is better to design schemes in the standard model, without any random oracle [BR93], and proven secure under a classical and well-studied security assumption.

The two expected security properties are the *unforgeability*, protecting the signer, and the *blindness*, protecting the user (see Figure 4.1 and Figure 4.2).

— The *unforgeability* is the $\mathsf{EUF}-\mathsf{CMA}$ property states that a malicious user shouldn't be able to compute $n+1$ valid signatures on different messages after at most n interactions with the signer.

— The *blindness* property says that a malicious signer who signed two messages m_0 and m_1 shouldn't be able to decide which one was signed first.

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\mathfrak{K})$
1. $(\mathsf{param}) \leftarrow \mathsf{BSSetup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{BSKeyGen}(\mathsf{param})$
3. For $i = 1,\ldots,q_s$, $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}),\mathcal{A}(\texttt{INIT} : \mathsf{vk})\rangle$
4. $((m_1,\sigma_1),\ldots,(m_{q_s+1},\sigma_{q_s+1})) \leftarrow \mathcal{A}(\texttt{GUESS} : \mathsf{vk})$;
5. `IF` $\exists i \neq j, m_i = m_j$ `OR` $\exists i, \mathsf{Verify}(\mathsf{vk}, m_i, \sigma_i) = 0$
   `RETURN` 0
6. `ELSE RETURN` 1

Figure 4.1 – Unforgeability Game for a $\mathcal{BS}$ Scheme

$\mathsf{Exp}^{\mathsf{bl}-b}_{\mathcal{BS},\mathcal{S}^*}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{BSSetup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk}, m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{param})$
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. $b^* \leftarrow \mathcal{S}^*(\texttt{GUESS} : m_0, m_1)$;
6. `RETURN` $b^* = b$.

Figure 4.2 – Blindness Game for a $\mathcal{BS}$ Scheme

4.1.3 Contribution

As said earlier, we proposed a new blind signature scheme, which is round-optimal, constant-size, in the standard model and with a classical assumption. Our construction

follows the framework presented by Fischlin [Fis06], and adapts an idea from [BFPV11] to decrease some cost.

The main tool used in our construction can be seen as a side contribution: we give the first signature scheme on randomizable ciphertexts [BFPV11] based on structure-preserving signatures, which are furthermore constant-size. To this aim, we prove that we can adapt the structure-preserving signatures proposed in [KPW15] to sign classes of elements, in the spirit of [HS14]. A class of elements is a group of elements which are ciphertexts of the same message. In other words, these schemes allow to give a signature on a ciphertext C, such that one can derive a signature on any randomization of this ciphertext. Of course, the unforgeability still guarantees that no adversary can generate a signature on an unsigned class, *i.e.* on a ciphertext without having seen a signature on a randomization of this ciphertext.

When comparing our schemes with existing ones (see Figure 4.3), one can see that we manage to keep all the communication constant-size with respect to the message length ℓ, while relying on a standard assumption.

Scheme	User	Server	Signature	# Rounds	Assumptions
[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
[BPV12b]	$\mathcal{O}(\ell/\log(\ell))\mathbb{G}_1$	$2\mathbb{G}_1, 1\mathbb{G}_2$	2 $\mathbb{G}_1, 1\mathbb{G}_2$	2	SXDH
[BBCF20]	$2\mathbb{G}_1$	$6\mathbb{G}_1, 1\mathbb{Z}_p$	$12\mathbb{G}_1, 8\mathbb{G}_2$	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 4.3 – Efficiency comparison of [BBCF20] with existing schemes in the standard model

4.2 Signature on Randomizable Ciphertext

A signature on randomizable ciphertexts, introduced in [BFPV11], is a scheme that allows a user to sign a ciphertext, as well as all its randomizations, plaintext included. It has been further generalized in [HS14, FHS19] to structure-preserving signature on equivalence classes.

4.2.1 Definition and Security Properties

Definition 13. *A Signature on Randomizable Ciphertexts (*SRC*) scheme is composed by the seven following algorithms:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$*: generates the global parameters* param *for the associated encryption and signature schemes.*
- $\mathsf{KeyGen}_E(\mathsf{param})$ *outputs a pair of keys, a (public) encryption key* ek *and a (private) decryption key* dk*;*

— $\mathsf{KeyGen}_S(\mathsf{param})$ *outputs a pair of keys, the (private) signature key* sk *and the (public) verification key* vk.

— $\mathsf{Enc}(\mathsf{ek}, \mathsf{vk}, m; r)$ *outputs a ciphertext* c *on message* $m \in \mathcal{M}$ *with* ek*, using the random coins* $r \in \mathcal{R}$.

— $\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; s)$*, with random coins* $s \in \mathcal{R}$*, outputs a signature* σ*, or* $\perp$ *if* c *is not valid (w.r.t.* ek*, and possibly* vk*).*

— $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{vk}, c)$ *decrypts* c *using* dk*. It outputs the plaintext, or* $\perp$ *if* c *is invalid w.r.t. the encryption key* ek *and possibly the verification key* vk.

— $\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, c, \sigma)$ *checks whether* σ *is a valid signature on* c*, w.r.t. the verification key* vk*. It outputs 1 if* σ *is valid and 0 in case of an invalid ciphertext under the encryption key* ek *or an invalid signature under the verification key* vk.

— $\mathsf{Random}(\mathsf{vk}, \mathsf{ek}, c, \sigma; r')$ *outputs a ciphertext* c' *that encrypts the same message as* c *under* ek*, and a signature* σ' *on* c'.

A signature on ciphertexts is called ciphertext randomizable *if a randomized signature on a randomized ciphertext is statistically indistinguishable from a fresh one.*

We will denote by 0_e the neutral element in $\mathcal{R}$ that keeps the ciphertexts unchanged after randomization.

Defined hereinafter is a similar primitive called Extracable Signatures on Randomizable Ciphertexts. To obtain it, we have to provide an additionial algorithm to extract signatures on messages from signatures on ciphertexts. The already defined notion of unforgeability for SRC follows from the standard unforgeability property for digital signatures scheme.

Extractable Signatures on Randomizable Ciphertexts

In addition to all algorithms defined for SRC, we need, for an Extractable Signatures on Randomizable Ciphertexts (ESRC) scheme, the following algorithm:

— $\mathsf{SEDecrypt}(\mathsf{dk}, \mathsf{vk}, \sigma)$, which is given a decryption key, a verification key and a signature, outputs a signature σ'.

Thus, if we have $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}_S(\mathsf{param}), m \in \mathcal{M}$, randomness $r \in \mathcal{R}, s \in \mathcal{R}$, $c = \mathsf{Enc}(\mathsf{ek}, \mathsf{vk}, m; r)$ and $\sigma = \mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; s)$, the output $\sigma' = \mathsf{SEDecrypt}(\mathsf{dk}, \mathsf{vk}, \sigma)$ is a valid signature on m under vk, that is, $\mathsf{Verify}_{\mathcal{S}}(\mathsf{vk}, m, \sigma')$ is true.

An ESRC scheme $\mathcal{SC}$ allows the following. First, a user can encrypt a message m and obtain a signature σ on the ciphertext c. From (c, σ), the owner of the decryption key can now not only recover the encrypted message m, but also a signature σ' on the *message* m, using the algorithm $\mathsf{SEDecrypt}$. The signature σ on the ciphertext c could thus be seen as an *encryption of a signature* on the message m: for extractable signatures on ciphertexts, encryption and signing can thus be seen as commutative (see Figure 4.4).

On this figure, one can easily see that $\mathsf{SEDecrypt} \circ \mathsf{Sign} \circ \mathsf{Enc} = \mathsf{Sign}_S$, assuring therefore some kind of commutativity between the signature and the encryption:

— a message m can be encrypted using random coins r (Enc);

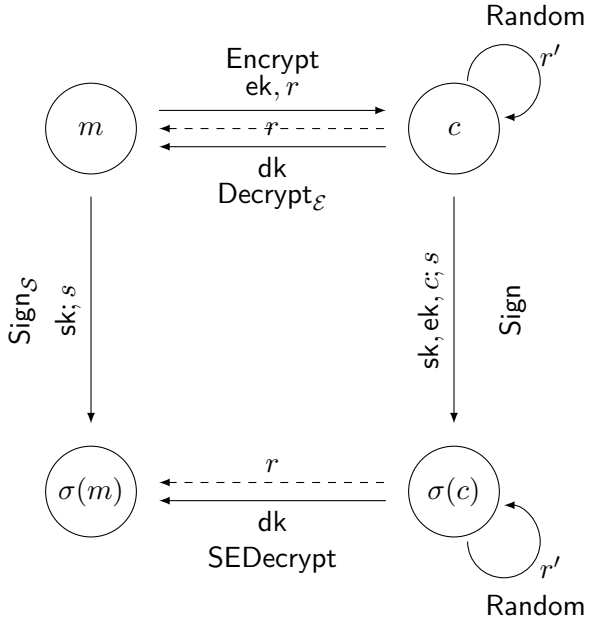

Figure 4.4 – Extractable signatures on randomizable ciphertexts

— the signer can sign this ciphertext (Sign) and anyone can randomize the inner cipher (Random);

— a signature on the plaintext can be obtained using either dk (for SEDecrypt) or the coins r (if $\sigma(c)$ has not been randomized); the result is the same as a signature of m by the signer (Sign$_S$).

Considering the randomness can be part of the group structure, using $\mathsf{Random}(\mathsf{vk}, \mathsf{dk}, c, \sigma; -r)$ can achieve the same as $\mathsf{SEDecrypt}(\mathsf{dk}, \mathsf{vk}, \sigma)$.

Practical aspect of Signature on Randomizable Ciphertext

In practice, when running Random, one wants to obtain a signature on a ciphertext unlinkable to the execution, besides the tag. At the end, the signature obtained may not have to be identical to a fresh signature nor to be randomizable again. Hence, we defined a new algorithm WRandom and its associated verification algorithm named WVerify.

— $\mathsf{WRandom}(\mathsf{vk}, \mathsf{ek}, c, \sigma; r')$ outputs a ciphertext c' that encrypts the same message as c under the public encryption key ek, and a signature σ' on c' valid under vk, with the same tag.

4.2.2 Instantiation with SXDH

Our construction is presented in Figure 4.5 and a generalization from any *mddh* assumption is presented in the next section. It should be noted, that in this case the WRandom algorithm drops the σ_{ek} component, while this may prevent a further ran-

domization of the signature, this is enough for the user to get a fresh c', σ' under the same tag.

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

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

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

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

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

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, c, \sigma)}$
Check whether:
$[\sigma_1\mathbf{A}^\top]_T = [(1, c^\top)\mathbf{C}^\top + \sigma_2(\mathbf{C}_0^\top + \sigma_\tau C_1^\top)]_T$
$[\sigma_{\mathsf{ek}}\mathbf{A}^\top]_T = [(0, \mathsf{ek}^\top)\mathbf{C}^\top + \sigma_2(\mathbf{C}_0^\top + \sigma_\tau C_1^\top)]_T$

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

$\underline{\mathsf{WRandom}(\mathsf{vk}, \mathsf{ek}, c, \sigma)}$:
$r' \xleftarrow{\$} \mathbb{Z}_p$
Compute $c' = c + [r', r'h]_1$ and update:
$\sigma_1' = \sigma_1 + \mathbf{r}'^\top\sigma_{\mathsf{ek}}$, $\sigma_2' = (1 + \mathbf{r}')\sigma_2$
Return c' and $\sigma' = (\sigma_1', \sigma_2', \sigma_\tau)$

$\underline{\mathsf{WVerify}(\mathsf{vk}, \mathsf{ek}, c', \sigma')}$
Check whether:
$[\sigma_1\mathbf{A}^\top]_T = [(1, c^\top)\mathbf{C}^\top + \sigma_2(\mathbf{C}_0^\top + \sigma_\tau C_1^\top)]_T$

Figure 4.5 – SXDH Instantiation of Constant Size SRC [BBCF20].

Correctness: First, show the correctness of signature σ. With definitions of $\mathbf{C}$, $\mathbf{C}_0$ and $\mathbf{C}_1$, we obtain:

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

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

Now, show the correctness of randomised signature σ':

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

Hence, our SRC is correct.

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

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

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

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

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

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

Figure 4.6 – Generic Construction of Constant Size SRC

4.3 SRC to Blind Signature

In this section, we present our blind signature based on the previous signature on randomizable ciphertext.

4.3.1 High-level Idea

Following [BFPV11], we can now provide a blind signature scheme using our freshly made signature scheme on randomizable ciphertexts.

The intuition is that after recovering the signature $\sigma_{\mathsf{SRC}} = \mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c = \mathsf{Enc}(\mathsf{ek}, \mathsf{vk}, m; r)); s)$, a user can compute $\sigma = \mathsf{WRandom}(\mathsf{vk}, \mathsf{ek}, c, \sigma_{\mathsf{SRC}}, -r)$, which is a valid signature on $(1, m)$. This gives him a signature on m.

The tricky part is now to achieve blindness. While fully randomizing the signature (as in [BFPV11]) would be the best solution, this is not possible with the signature from [KPW15] that we consider in this paper. Instead, following Fischlin's idea in [Fis06], we add a complete zero-knowledge proof of the knowledge of σ. Moreover, due to the fact that we cannot extract a scalar τ from a commitment, we need to commit to τ in $\mathbb{G}_2$, and to the original $\sigma_3 = [\tau\sigma_2]_1$.

A high-level idea of the process (pre-blinding) is explained in the figure 4.7.

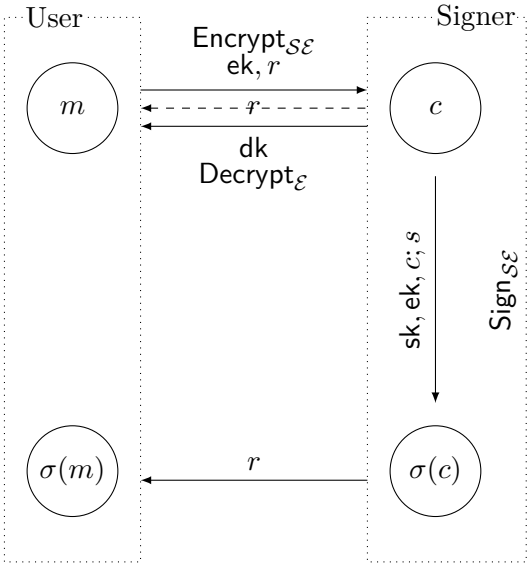

Figure 4.7 – High level Blind Signature

4.3.2 Overview of the construction

In this section, we instantiate algorithms from the definition 12 of a blind signature $\mathcal{BS} = (\mathsf{BSSetup}, \mathsf{BSKeyGen}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Verify})$.

— $\mathsf{BSSetup}(\mathfrak{K})$ calls the setup algorithm of the SRC scheme. It outputs param.

— $\mathsf{BSKeyGen}(\mathsf{param})$ calls the KeyGen_S algorithm of the SRC scheme. It is run by the server $\mathcal{S}$.

— Signature Issuing is described by Figure 4.8. It is an interactive protocol in which the user $\mathcal{U}$ has to encrypt its message and send it to the signer $\mathcal{S}$ which in return computes a SRC signature σ_{SRC}. Having received σ_{SRC}, $\mathcal{U}$ checks its validity using vk. If it is valid, he extracts a signature on m by using the WRandom algorithm.

Following Fischlin's framework, the blind signature consists in the NIZK proof π guaranteeing $\mathcal{U}$ has the elements satisfying the signature verification equations w.r.t to vk and m.

— $\mathsf{Verify}(\mathsf{vk}, \sigma)$ calls the NIZK proof verification algorithm. If the proof is valid, the signature is valid and it outputs 1, otherwise 0.

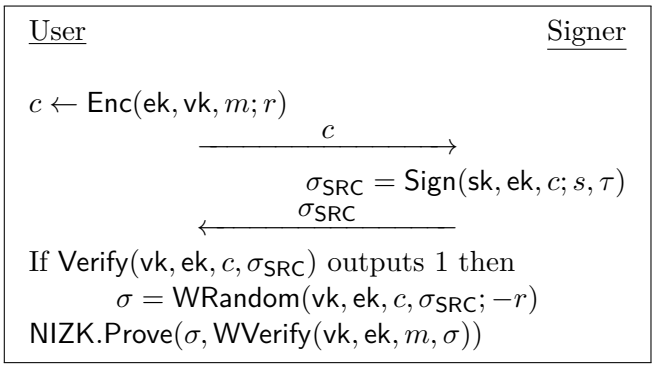

Figure 4.8 – Blind Signature

4.3.3 Security Results and Proofs Ideas

Blindness

Theorem 1. *Our construction achieves the blindness property under the* SXDH *assumption.*

Proof idea: Every value sent by the user is either encrypted (committed) or a Zero-Knowledge proof. The flow in the SRC hides the target message, while the Zero-Knowledge proof hides both the tag and the randomness used in the signature. Hence, under the IND-CPA property of the encryption and the Zero-Knowledge property of the proof, the scheme achieves blindness.

Proof. We proceed via a series of games and we use Adv_i to denote the advantage of $\mathcal{A}$ in Game i.

We present this different games in the following Figure (4.9).

Game 0 This is the Blindness experiment from figure 4.2: $\mathsf{Adv}_0 = \mathsf{Adv}^{Blind}_{\mathcal{BS}}$.

In this game, we don't modify existing algorithms.

During the different games, $\mathcal{A}$ proposed two messages m_0 and m_1 to the challenger.

Game 1 Both proofs are replaced by simulated ones thanks to the Zero-Knowledge property of the Groth-Sahai proof system as we can see in the Figure 4.9, We replace the NIZK algorithm by a simulated one. Thus, we have:

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

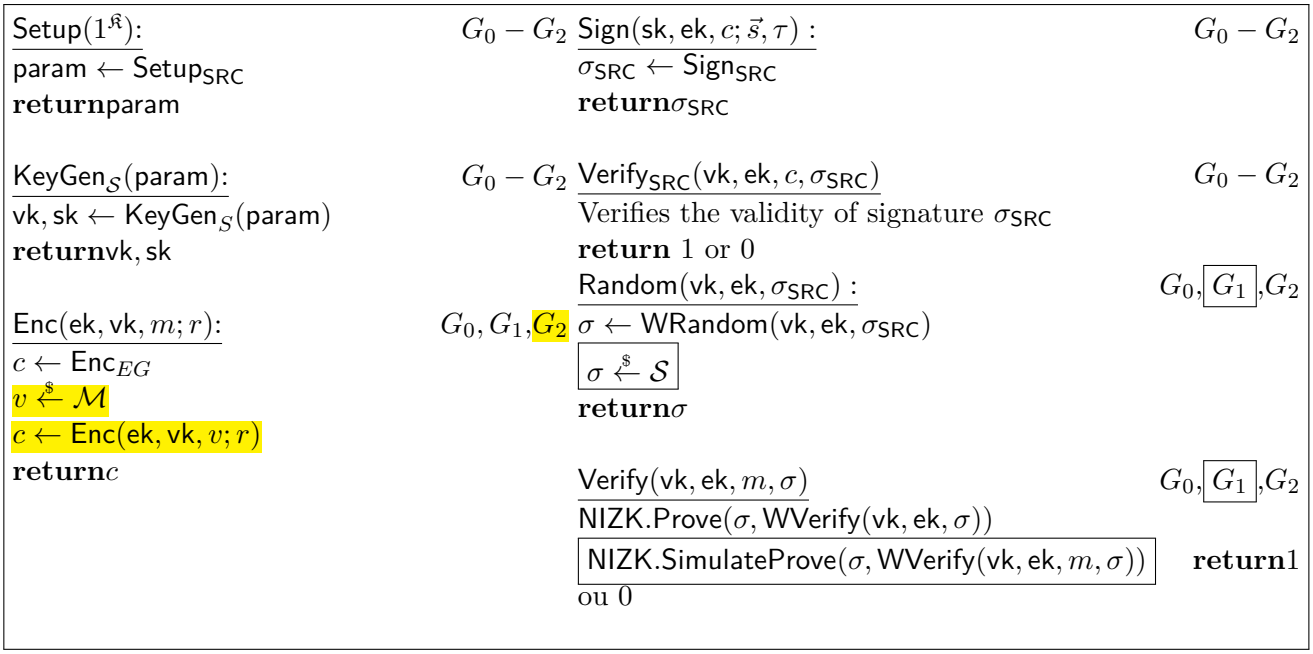

Figure 4.9 – Security Games for the blindness.

Game 2 Both encryptions of m_0 and m_1 are replaced by random group elements *i.e* the challenger encrypts a random group element noted v instead of m_b. By the indistinguishability property of the ElGamal encryption scheme, we have:

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

In this game, $\mathcal{A}$ is given absolutely no information on m_0 and m_1 therefore $\mathsf{Adv}_2 = 0$. □

Unforgeability

In order to prove the notion of unforgeability for this instantiation, we need the following lemma from [KPW15]. Compared to the original signature, we add a game in order to randomize σ_{ek}.

Lemma 1 (Computational core lemma for unbounded CMA-security)**.** *For all adversaries $\mathcal{A}$, there exists a challenger $\mathcal{B}$ with $\mathcal{T}(\mathcal{A}) \approx \mathcal{T}(\mathcal{B})$ and*

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

where :

— $\mathcal{O}(\tau)$ *returns* $([b\mu\mathbf{a}^{\perp}+\mathbf{r}^{\top}(\mathbf{P}_0+\tau\mathbf{P}_1)]_1, [\mathbf{r}^{\top}\mathbf{B}]_1) \in (\mathbb{G}_1^{1\times(k+1)})^2$ *with* $\mu \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^k$ *and adds* τ *to* $\mathcal{Q}_{msg}$. *Here,* $\mathbf{a}^{\top}$ *is a non-zero vector in* $\mathbb{Z}_p^{1\times(k+1)}$ *that satisfies* $\mathbf{a}^{\perp}\mathbf{A} = 0$.

— $\mathcal{O}^*([\tau^*]_2)$ *returns* $[\mathbf{K}_0 + \tau^*\mathbf{K}_1]_2$. $\mathcal{A}$ *only gets a single call* τ^* *to* $\mathcal{O}^*$.

— $\mathcal{Q}$ *is the number of queries* $\mathcal{A}$ *makes to* $\mathcal{O}_b$.

Theorem 2. *Our construction is unforgeable under the* XDH *in* $\mathbb{G}_1$ *and* $\mathcal{D}_1 - \mathsf{KerMDDH}$ *in* $\mathbb{G}_2$.

Following the original proof from [KPW15], we proceed via successive games. First, we guess whether the adversary picks a fresh tag for the forgery or reuse one of the signature he received, and we simulate all the other answers. Then, we show that for a given tag, the signature becomes a valid 1-time signature.

Proof. **Game 1.** In this game, we modify the verification algorithm. The pairing equation became : $[\sigma_1 \cdot 1]_T = [(1, \mathbf{m}^{\top}\mathbf{K}) \cdot 1 + \sigma_2 \cdot (\mathbf{K}_0 + \tau\mathbf{K}_1)]_T$.

Suppose $[\sigma_2 \cdot \sigma_4]_T = [\sigma_3]_T$, we note that:

$$\begin{aligned}
&[\sigma_1 \cdot \mathbf{A}]_T = [(1, \mathbf{m}^{\top}) \cdot \mathbf{C} + \sigma_2 \cdot \mathbf{C_0} + \sigma_3 \cdot \mathbf{C_1}]_T \\
\iff &[\sigma_1 \cdot \mathbf{A}]_T = [(1, \mathbf{m}^{\top}) \cdot \mathbf{KA} + \sigma_2 \cdot \mathbf{K}_0\mathbf{A} + \sigma_3 \cdot \mathbf{K}_1\mathbf{A}]_T \\
\iff &[\sigma_1 \cdot 1]_T = [(1, \mathbf{m}^{\top}) \cdot \mathbf{K} + \sigma_2 \cdot \mathbf{K}_0 + \sigma_3 \cdot \mathbf{K}_1]_T \\
\iff &[\sigma_1 \cdot 1]_T = [(1, \mathbf{m}^{\top}) \cdot \mathbf{K} + \sigma_2 \cdot (\mathbf{K}_0 + \tau\mathbf{K}_1)]_T
\end{aligned}$$

The value $\sigma_1 - ([(1, \mathbf{m}^{\top})\mathbf{K}]_1 + \sigma_2\mathbf{K}_0 + \sigma_3\mathbf{K}_1) \in \mathbb{G}_1^{1\times(k+1)}$ is a non-zero vector in the kernel of $\mathbf{A}$, which is hard to be computed under the $\mathcal{D}_k$-KerMDDH assumption in $\mathbb{G}_2$. This means that: $|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \mathsf{Adv}^{k-\mathsf{MDDH}}_{\mathcal{D}_k,\mathsf{Setup}}(\mathcal{B}_0)$.

Game 2. Let $\tau_1, ..., \tau_{\mathcal{Q}}$ the randomly chosen tags in the $\mathcal{Q}$ queries made by the adversary $\mathcal{A}$ to OSign. We abort if $\tau_1, ..., \tau_{\mathcal{Q}}$ are not all distinct. So, we obtain: $\mathsf{Adv}_1 \geq \mathsf{Adv}_1 - \mathcal{Q}^2/2q$.

Game 3. We define $\tau_{\mathcal{Q}+1} \stackrel{\mathsf{def}}{=} \tau^*$. Now, pick $ß^* \xleftarrow{\$} [\mathcal{Q}+1]$ and abort if i^* is not the smallest index of i^* for which $\tau^* = \tau_i$. In the rest of the proof, we focus on the case we do not abort *i.e* $\tau^* = \tau_{i^*}$ and $\tau_1, ..., \tau_{i^*-1}$ are all different from τ^*. Given τ, OSign can check whether τ^* equals τ. For the rest $i^* - 1$ queries, answer NO, and starting from the i^*'th query, we know τ^*.

Hence, we have:

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

Game 4.1. We switch OSign to OSign_1^*:

$\underline{\mathsf{OSign}_1^*}$:	OSign_2^*
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p; \tau \xleftarrow{\$} \mathbb{Z}_p; \mu_1 \xleftarrow{\$} \mathbb{Z}_p$	
$\sigma_1 \stackrel{\mathsf{def}}{=} [(1, C_1, C_2)\mathbf{K} + \mu_1 \mathbf{a}^\perp + \mathbf{r}^\top(\mathbf{P_0} + \tau \mathbf{P_1})]_1 \in \mathbb{G}_1^{1\times 2}$	
$\sigma_{\mathsf{ek}} \stackrel{\mathsf{def}}{=} [(0, 1, \mathsf{ek})\mathbf{K} + \mathbf{r}^\top(\mathbf{P_0} + \tau \mathbf{P_1})]_1 \in \mathbb{G}_1^{1\times 2}$	
$\sigma_{\mathsf{ek}} \stackrel{\mathsf{def}}{=} [(0, 1, \mathsf{ek})\mathbf{K} + \mu_2 \mathbf{a}^\perp + \mathbf{r}^\top(\mathbf{P_0} + \tau \mathbf{P_1})]_1 \in \mathbb{G}_1^{1\times 2}$	
$\sigma_2 \stackrel{\mathsf{def}}{=} [\mathbf{r}^\intercal \mathbf{B}^\intercal]_1 \in \mathbb{G}_1^{1\times 2}$, $\sigma_\tau \stackrel{\mathsf{def}}{=} \tau \in \mathbb{Z}_p$	
$\mathtt{Return}(\sigma_1, \sigma_{\mathsf{ek}}, \sigma_2, \sigma_\tau)$	

Here, $\mathbf{a}^\perp \in^{1\times(k+1)}$ is non-zero vector in the kernel of $\mathbf{A}$ such that: $\mathbf{a}^\perp \mathbf{A} = 0$. We will apply Lemma 1 in order to show: $|\mathsf{Adv}_3 - \mathsf{Adv}_{4.1}| \geq 2\mathcal{Q}\mathsf{Adv}^{mddh}_{\mathcal{D}_k,\mathsf{Setup}}(\mathcal{B}_1) + \mathcal{Q}/q$. We pick $\mathbf{K}$ and we use $\mathcal{O}_b$ to simulate either OSign or OSign_1^* and $\mathcal{O}^*$ to simulate Verify as follow:

— For the i'th signing query where $i \neq i^*$, we query $\mathcal{O}_b$ at $\tau \xleftarrow{\$} \mathbb{Z}_p$ to obtain:

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

with $b \xleftarrow{\$} \{0, 1\}$ and we return:

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

— For the i^*'th query, where $i^* \leq \mathcal{Q}$, we run Sign honestly.

— For Verify^*, we will query $\mathcal{O}^*$ on τ^* to get $[\mathbf{K}_0 + \tau^* \mathbf{K}_1]_2$. The latter is sufficient to simulate Verify^* query by computing $[\sigma \cdot (\mathbf{K}_0 + \tau^* \mathbf{K}_1)]_T$.

So, we are allowed to build a distinguisher for Lemma 1.

Game 4.2

We have to modify σ_{ek} in the same way as before:

In order to apply lemma 1, we simulate oracles:

— For $i \neq i^*$, we have:

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

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

— For the challenge, we run Sign honestly.

Game 5. In this game we modify the matrix $\mathbf{K}$ in Setup algorithm. We switch $\mathbf{K}$ by $\mathbf{K} = \mathbf{K}' + \mathbf{u}\mathbf{a}^\perp$ with $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{(n+1)\times(k+1)}$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{n+1}$. $\mathbf{u}\mathbf{a}^\perp$ is masked by a uniform matrix $\mathbf{K}'$, $\mathbf{K}$ is still uniformly random. Thus, Game 5 and Game 4.2 are identical, so we have: $\mathsf{Adv}_5 = \mathsf{Adv}_{4.2}$.

Now, we have to bound Adv_5.

— $C = \mathbf{K}\mathbf{A} = (\mathbf{K}' + \mathbf{u}\mathbf{a}^\perp)\mathbf{A} = \mathbf{K}'\mathbf{A}$. $\mathbf{u}$ is completely hide.

— OSign_2^* on $(C_1, C_2, \mathsf{ek}, \tau)$ for $\tau \neq \tau^*$ returns: $(1, C_1, C_2)(\mathbf{K}' + \mathbf{u}\mathbf{a}^\perp) + \mu_1 \mathbf{a}^\perp$ and $(0, 1, \mathsf{ek}) + (\mathbf{K}' + \mathbf{u}\mathbf{a}^\perp) + \mu_2 \mathbf{a}^\perp$. $\mathbf{u}$ is hidden since outputs are identically distributed as: $(1, C_1, C_2)\mathbf{K}' + \mu_1 \mathbf{a}^\perp$ and $(0, 1, \mathsf{ek})\mathbf{K} + \mu_2 \mathbf{a}^\perp$.

— OSign_2^* on τ^* leaks $(1, C_1, C_2)\mathbf{K}' + \mu_1\mathbf{a}^\perp$ and $(1, \mathsf{ek})\mathbf{K} + \mu_2\mathbf{a}^\perp$, which is captured by $(1, C_1, C_2)\mathbf{u}$ and by $(0, 1, \mathsf{ek})\mathbf{u}$.

The adversary has to compute correctly: $(1, C_1^*, C_2^*)(\mathbf{K}' + \mathbf{u}\mathbf{a}^\perp)$ and $(0, 1, \mathsf{ek})(\mathbf{K}' + \mathbf{u}\mathbf{a}^\perp)$ to convince Verify^* to accept a signature σ^* on $(1, C_1^*, C_2^*)$ and on $(0, 1, \mathsf{ek}^*)$. As $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, $(1, C_1^*, C_2^*)\mathbf{u}$ and $(0, 1, \mathsf{ek}^*)\mathbf{u}$ are in $\mathbb{Z}_p$.

Given $(1, C_1^*, C_2*)\mathbf{u}$ and $(0, 1, \mathsf{ek}^*)$for any adaptively chosen $(C_1^*, C_2^*) \neq (C_1, C_2)$ and $\mathsf{ek}^* \neq \mathsf{ek}$, $(1, C_1^*, C_2^*)\mathbf{u}$ and $(0, 1, \mathsf{ek}^*)\mathbf{u}$ are uniformly random over $\mathbb{Z}_p$ from the adversary's view-point. □

Special Optimization in Asymmetric Settings

In the asymmetric settings, it is interesting to note that we do not have to hide both σ_2 and σ_3. Because, an adversary (the signer) knowing $s, \tau \in \mathbb{Z}_p, [r]_1$ cannot distinguish $\sigma_2' = [(1-r)s\vec{B}]$ from a fresh term.

4.4 Application to e-voting

In voting schemes, anonymity is a crucial property, since nobody should be able to learn the content of someone else's vote. This can be achieved using homomorphic encryption schemes. Basically, each user will compute his vote v_i, commit it into c_i, and then, through the homomorphic property of the encryption scheme, the voting center will compute $f(c_1, \ldots, c_n)$ and open it to solely obtain the global result of the election.

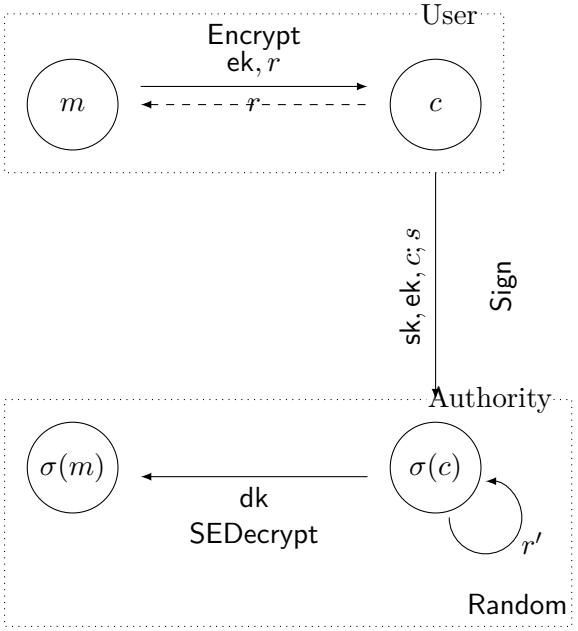

Figure 4.10 – Viewing an SRC as a e-voting solution

Nonetheless, with this technique remains the issue of *vote sellers*: a voter may sell his vote to a buyer. To prove his vote, he merely has to reveal the random coins used when encrypting his vote; the buyer can verify that the claimed vote is exactly the same by encrypting it again. There is a way to avoid that: the voting center randomizes c into c' and then proves, using a designated-verifier zero-knowledge proof, that c and c' contain the same vote. Finally, after receiving c' and being convinced by the proof, the voter signs c'. This way, the signature is on c' for which the voter does not possess the random coins anymore.

Our SRC allows to avoid those extra interactions: a voter simply creates a ciphertext c containing his vote v and produces a signature σ on c. The authority is now able to randomize both ciphertext c and signature σ as c' and σ' so that the random coins used in c' is unknown to the signer. The unforgeability notion guarantees that the signature σ' is valid *w.r.t.* the encryption key ek and the verification key vk making modification to the vote v impossible. Thus, we have built a non-interactive *receipt-free* voting scheme. [1]

Since our SRC candidates use both randomizable and homomorphic encryption schemes, classical techniques for voting schemes with homomorphic encryption and threshold decryption can be used [BFP+01]: there is no risk for the signature on the ciphertext to be converted into a signature on the plaintext if the vote authority uses the decryption capability on the encrypted tally only.

The size of the ballot is only 6 group elements and a scalar in the instantiation with using ElGamal, independently from the number of check boxes in the vote.

1. We want to stress, that as the secret signing key is **secret**, users cannot craft fake encrypted ballot in a way that would allow two of them to be combined in a third one. In addition, as the signer changes the tag with every signing query, this combination would be impossible.

Chapter 5

Efficient and Round-Optimal Secret Handshake

Secret handshakes are surrounded by an aura of mystery, with images of members of secret societies wanting to exchange confidential information. Such protocols enable two parties to authenticate privately to each other, with the means of credentials given by certification authorities (ensuring they belong to the same organization, for instance). The purpose of these schemes is two-fold, in the sense that they are supposed to provide a meaningful way for two members of an organization to identity each other, while ensuring that no one outside the organization can neither impersonate an organization member nor recognize one.

Secret handshakes can be seen as a special kind of authenticated key exchange: In those protocols, two users possessing a credential should be able to derive a shared key if and only if their credentials were delivered by the same authority. Outsiders (including the authority, or members of the same agency) should neither be able to determine whether the protocol was successful, nor compute the final key.

Contents

5.1 General Overview

In the digital world, applications of secret handshakes can cover a large area ranging from ad-hoc network to smart contract in service oriented architecture (SOA). Ad-hoc networks and more generally peer-to-peer communications are a transposition from the real world use case. They offer opportunistic communications which can use secret handshakes to enhance the privacy of involved parties without revealing information to outsiders. Indeed, privacy is becoming a central feature and secret-handshakes are an interesting solution to enhance existing applications especially when it comes to communications. For instance, there are applications like Signal or Telegram that provide secure communications between communicating parties thanks to end-to-end encryption. But, they rely on a central server that is in charge of message routing and so has a full knowledge of origin and destination.

To overcome this limitation, recent solutions like MASHaBLE [MNL16] propose to use secret-handshakes to build communities in a secure way without leaking information to a central server or to anyone outside of the community.

In SOA, secret handshakes can be envisioned as part of a privacy-preserving key agreement protocol to allow a client to connect to a service from the same organization as him for instance. Indeed, they provide a secure, unobservable and untraceable solution for privacy-preserving mutual authentication.

Since their introduction in [BDS+03a], secret handshakes have been widely studied by the cryptographic community [CJT04, TX06, AKB07, CKSK08, BBC+13a]. While the earlier papers present plenty use cases of secret handshakes, they lack a proper *solid* security model. On the contrary, the authors of [BBC+13a] propose a UC-secure scheme with a proper definition of ideal functionality, but the resulting instantiation lacks any trait of usability.

5.2 Related Work

In the seminal work of Balfanz, Durfee, Shankar, Smetters, Staddon and Wong [BDS+03b], the authors present the concept of secret handshake protocol as a way to allow a user to authenticate itself to a server without any leak of information regarding the credentials. They also provide a 3-round construction secure in the random oracle model (ROM) and which supports authentication with different roles in a same group (that is to say than an authentication happens only if both users are in the same group and the roles of their counterpart is adequate). In [CJT04], secret handshakes are created through a primitive called CA-Oblivious Transfer; Their construction is computationally cheaper but offers the same functionalities, and the security still relies on the ROM.

Xu and Yung, in [XY04], propose a secret handshake protocol with *unlinkability*. A user can execute the protocol multiples times with the same credentials without an

eavesdropper linking the transcripts to him. However, this privacy notion is limited as *unlinkability* is achieved through the use of k public keys of other members of the same group resulting in a computational overhead.

Secret handshakes are extended to the multi-party setting in [TX06] and [JKT06].

Atenesie, Blanton and Kirsch present in [AKB07] an efficient Secret handshake scheme with a stronger notion of *unlinkability* and the support of attributes. Its security relies on the BDH and SXDH assumptions.

Jarecki and Liu, in [JL07], also present a secret handshake scheme with *unlinkability* without single-use credentials; As a consequence, it supports revocation and no leakage of information. In addition, they offer strong definitions of *security* and *privacy* for secret handshake (SH) schemes.

The same authors, in [JL09], provide a new and more efficient construction of SH schemes using a primitive they define as a Conditional Oblivious Transfer (COT). It allows them to have a 3-round SH protocol in the ROM ($\mathcal{O}(1)$ otherwise). Their scheme can also be extended to a *private envelope* scheme that enables a receiver to recover an encrypted message if and only if its certificate matches the sender's authorization policy, respecting a similar *privacy* property as in a secret handshake.

Many other constructions were proposed with added features such as user revocation or relying on different security assumptions.

5.2.1 Contribution

With this work, our goal is to bridge the gap between those two worlds by formalizing a specific security model derived from the classical models for Password-based Authenticated Exchange [BPR00, BBC+13b]. This model is aimed at providing additional security guarantees (with respect to the authority issuing the keys, for instance) than the classical designs in existing papers, while ensuring efficient instantiability. Furthermore, our model can handle *Extended* Secret Handshake, where two users (not necessarily belonging to the same organization) can authenticate each other as long as they possess a credential delivered by the authority that the other user is expecting. This encompasses the original definition, as a user can expect to interact with someone of the same organization, *i.e.* owning a credential issued by the same authority than himself.

We then propose a generic framework to construct round-optimal secret handshake protocols. This framework relies on three main tools:

— *Digital Signatures* [DH76, GMR88]: They allow a signer to authenticate some string. In our cases, each organization authority would have a public verification key [1] and a secret signing key that they would use to sign each member's identity. Each member would then receive a (valid) signature on his identity by his (respective) organization authority.

1. Technically, one could assume that each organization public key is only given to members of the organization, but to model potential corruption, it is safer to assume that even outsiders may know them.

— *CCA-2 Encryption* [CS02]: As a public signature would leak a user's associated organization, we need to use a way to mask part of those signatures. The easiest way is to encrypt them. As we want to be able to handle several sessions, the CCA-2 property is here to prevent adversarial reuse of flows.

— *Smooth Projective Hash Functions*: Assume there exists inside a set a *language*, *i.e.* a subset verifying the hard subset membership problem (given a word in the whole set, it is difficult to decide whether this word is in the language or not). Words in this particular language are associated with a *witness* of this membership. Here, CCA-2 ciphertexts of a signature will define such a language in the set of tuples (the randomness used in the encryption being the witness). Then, a smooth projective hash function allows to compute a hash value (to be used as a shared key between two parties) in two different ways: Either from a public ciphertext and a secret hashing key, or from the private signature and randomness and a public (projected) hashing key.

Next, we give two concrete instantiations of our framework. The first one relies on the BLS signatures [BLS04]. This is mostly a toy example, as it provides an example (only) provable in the Random Oracle Model with a construction close to the initial one [BDS+03a]. This example allows to see our framework in action, and leads to a very efficient instantiation provable in the revisited security model. Our second example uses the Structure-Preserving (SP) Signature defined by Kiltz *et al.* [KPW15]. This signature is more convoluted as its initial goal was to be structure-preserving (messages, keys and signatures are all group elements). With this design, we manage to achieve a constant-size secret handshake protocol in the standard model (in particular, without any random oracle). We then propose further improvements for efficiency. While classical definitions consider signatures over an identity, standard use cases consider credentials authenticating a user public key. Thus, using a structure-preserving signature would allow to preserve some possibly needed algebraic properties.

Finally, we provide an efficiency analysis, showing that our implementations are efficient and could already be used in a real-life scenario, such as network protocols (TLS, SSH). An important property is that our secret handshake schemes are also *round-optimal* in the sense that both users only need to send their data once, and that the flows can be concurrent (*i.e.* sent at the same time without waiting for the other to be delivered).

5.3 Model

In this section, we give the formal definition of secret handshake and present the security model in which we prove our schemes.

Definition of Secret Handshake. In a bi-directional private authentication, also called (unlinkable) secret handshake, introduced in 2003 by Balfanz *et al.* [BDS+03b] (see also [JL09, AKB07]), two parties want to authenticate to each other as certified by given certification authorities (proving they are affiliated to given groups, for instance).

This authentication is mutually private, in the sense that it leaks no information on a party to someone who does not satisfy this party's authentication policy. In particular, the certifications of a party are hidden. In case of an unlikable secret handshake, protocol instances involving the same participant are unlinkable. In any case, an eavesdropper does not learn anything on the participants, even the success or failure of the protocol. At the end of the execution of the protocol, the parties can share a session key which will secure further communication between them.

We now give a simple definition of a secret handshake scheme $\mathcal{SH}$ which allows two users of a same group to exchange secrets. Those users belong to a group created by a group administrator (GA) and can execute the protocol successfully only among them. Thus, the $\mathsf{Handshake}$ algorithm described below results in the users id_1 and id_2 sharing a common key if they belong to the same group $\mathcal{G}$.

— $\mathsf{Setup}(1^{\mathfrak{K}})$ generates the parameters of the scheme;

— $\mathsf{CreateGroup}(\mathcal{G})$ is run by an administrator to generate a group secret value $\mathcal{S}_{\mathcal{G}}$ associated with a group public value $\mathcal{P}_{\mathcal{G}}$;

— $\mathsf{UserEnrollment}(\mathsf{id}, \mathcal{G}, \mathcal{S}_{\mathcal{G}})$ is run by an administrator to create, using a secret $\mathcal{S}_{\mathcal{G}}$, a secret $\mathcal{S}_{\mathsf{id},\mathcal{G}}$ that is given to user id in group $\mathcal{G}$. This secret value represents its enrollment in group $\mathcal{G}$;

— $\mathsf{Handshake}((\mathsf{id}_1, \mathcal{S}_{\mathsf{id}_1,\mathcal{G}_1}), (\mathsf{id}_2, \mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2}))$ describes the handshake protocol executed by users id_1 (with secret $\mathcal{S}_{\mathsf{id}_1,\mathcal{G}_1}$) and id_2 (with secret $\mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2}$);

Revisited Model. The existing security models for secret handshakes are not strong enough since some of them allow the group authority to detect if an execution of the protocol succeeds and others are restricted to a unique group.

A security model for secret handshakes in the universal composability framework [Can01] can be derived from the model defined for LAKE (language-authenticated key exchange) in [BBC+13a], as an extension of the UC security for PAKE [CHK+05]. An extension of the PAKE security model presented by Bellare, Pointcheval, and Rogaway [BPR00] has been given for LAKE in [BBC+13b] and could be derived for secret handshakes as well.

In this paper, we propose a simpler variant of these models from [BPR00, BBC+13b] for (unlinkable) secret handshakes with strong security properties. A secret handshake scheme needs to fulfill three desirable properties:

— *Correctness*: A protocol execution between two users of the same group always results in a key agreement. This means that the execution of the algorithm $\mathsf{Handshake}((\mathsf{id}_1, \mathcal{S}_{\mathsf{id}_1,\mathcal{G}_1}), (\mathsf{id}_2, \mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2}))$ results in id_1 and id_2 sharing a key as soon as $\mathcal{S}_{\mathsf{id}_1,\mathcal{G}_1} = \mathcal{S}_{\mathsf{id}_1,\mathcal{G}}$ and $\mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2} = \mathcal{S}_{\mathsf{id}_2,\mathcal{G}}$ are valid secret values representing the enrollments of id_1 and id_2 in group $\mathcal{G}$.

— *Impersonator resistance*: Even after having seen several interactions of the form $\mathsf{Handshake}((\mathsf{id}_1, \mathcal{S}_{\mathsf{id}_1,\mathcal{G}_1}), (\mathsf{id}_2, \mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2}))$ between users (id_1 potentially being corrupted), some of them being successful, an adversary not belonging to the correct organization (corresponding to the group $\mathcal{G}$) should not be able to successfully run the protocol and obtain a shared key with an honest user belonging to the group $\mathcal{G}$.

— *Privacy*: Any outsider (even the group authority) should not be able to detect if a protocol succeeds and/or guess to which organization belong the involved user id_1 after having seen an execution
$\mathsf{Handshake}((\mathsf{id}_1, \mathcal{S}_{\mathsf{id}_1,\mathcal{G}_1}), (\mathsf{id}_2, \mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2}))$ with another user id_2.

An unlinkable secret handshake scheme further fulfills the following property:

— *Unlinkability*: No adversary can link several transcripts to an honest user, even if the latter executed the protocol $\mathsf{Handshake}((\mathsf{id}_1, \mathcal{S}_{\mathsf{id}_1,\mathcal{G},i}), (\mathsf{id}_i, \mathcal{S}_{\mathsf{id}_i,\mathcal{G}}))$ multiple times with the same credentials $\mathcal{S}_{\mathsf{id}_1,\mathcal{G},i} = \mathcal{S}_{\mathsf{id}_1,\mathcal{G}}$.

$\mathsf{Exp}^{\mathsf{impres}}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K})$ [Impersonator resistance]
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $\mathcal{G} \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{param})$
3. $(\mathcal{S}_{\mathcal{G}}, \mathcal{P}_{\mathcal{G}}) \leftarrow \mathsf{CreateGroup}(\mathcal{G})$
4. $\forall i = 1...n \quad K_i \leftarrow \mathcal{O}_{\mathsf{handshake}}(\mathcal{A}(\texttt{INIT} : \mathcal{P}_{\mathcal{G}}), \mathcal{O}_{\mathsf{user}}((\mathcal{G}, \mathcal{P}_{\mathcal{G}}), (\cdot, \cdot)))$
5. $(K_{n+1}, \mathsf{transcript}) \leftarrow \mathcal{A}(\texttt{GUESS} : \mathcal{P}_{\mathcal{G}})$
6. $\texttt{RETURN} \ \ \mathcal{O}_{\mathsf{valid}}((\mathcal{G}, \mathcal{S}_{\mathcal{G}}), (K, \mathsf{transcript}))$

$\mathsf{Exp}^{\mathsf{privacy}-b}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K})$ [Privacy]
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathcal{G}_0, \mathcal{G}_1, \mathsf{id}_1, \mathsf{id}_2) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{param})$
3. $\forall i \quad (\mathcal{S}_{\mathcal{G}_i}, \mathcal{P}_{\mathcal{G}_i}) \leftarrow \mathsf{CreateGroup}(\mathcal{G}_i)$
4. $\forall (i,j) \quad \mathsf{UserEnrollment}(\mathsf{id}_j, \mathcal{G}_i, \mathcal{S}_{\mathcal{G}_i})$
5. $\mathsf{transcript}_b \leftarrow \mathcal{O}_{\mathsf{handshake}}(\mathcal{O}_{\mathsf{user}}((\mathcal{G}_b, \mathcal{P}_{\mathcal{G}_b}), (\mathsf{id}_1, \cdot)), \mathcal{O}_{\mathsf{user}}((\mathcal{G}_b, \mathcal{P}_{\mathcal{G}_b}), (\mathsf{id}_2, \cdot)))$
6. $\mathsf{transcript}_{1-b} \leftarrow \mathcal{O}_{\mathsf{handshake}}(\mathcal{O}_{\mathsf{user}}((\mathcal{G}_{1-b}, \mathcal{P}_{\mathcal{G}_{1-b}}), (\mathsf{id}_1, \cdot)), \mathcal{O}_{\mathsf{user}}((\mathcal{G}_{1-b}, \mathcal{P}_{\mathcal{G}_{1-b}}), (\mathsf{id}_2, \cdot)))$
7. $b^* \leftarrow \mathcal{A}(\texttt{GUESS} : \mathsf{id}_1, \mathsf{id}_2)$
8. $\texttt{RETURN} \ \ b^* = b$

$\mathsf{Exp}^{\mathsf{unlink}-b}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K})$ [Unlinkability]
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathcal{G}, \mathcal{P}_{\mathcal{G}}, \mathsf{id}, S_{\mathsf{id},\mathcal{G},0}, S_{\mathsf{id},\mathcal{G},1}) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{param})$
3. $K_b \leftarrow \mathsf{Handshake}((\mathcal{A}, \mathcal{S}_{\mathcal{A},\mathcal{G}}), \mathcal{O}_{\mathsf{user}}((\mathcal{G}, \mathcal{P}_{\mathcal{G}}), (\mathsf{id}, \mathcal{S}_{\mathsf{id},\mathcal{G},b})))$
4. $K_{1-b} \leftarrow \mathsf{Handshake}((\mathcal{A}, \mathcal{S}_{\mathcal{A},\mathcal{G}}), \mathcal{O}_{\mathsf{user}}((\mathcal{G}, \mathcal{P}_{\mathcal{G}}), (\mathsf{id}, \mathcal{S}_{\mathsf{id},\mathcal{G},1-b})))$
5. $b^* \leftarrow \mathcal{A}(\texttt{GUESS} : S_{\mathsf{id},\mathcal{G},0}, S_{\mathsf{id},\mathcal{G},1})$
6. $\texttt{RETURN} \ \ b^* = b$

Figure 5.1 – Security games for secret handshake schemes

These security notions can be formalized by security games in the Find-then-Guess

paradigm [BR94]. In these games, the adversary keeps some internal state between the various calls `INIT`, `FIND`and `GUESS`she can make:

— With $\mathcal{A}(\texttt{INIT} : \cdot)$, the adversary can choose whether to make an honest user or herself execute the protocol (see below).

— With $\mathcal{A}(\texttt{FIND} : \mathsf{param})$, she chooses the target elements to conduct her attack.

— With $\mathcal{A}(\texttt{GUESS} : \cdot)$, she outputs the result of her attack.

In order to reply to the adversary during these security games, the simulator makes use of the following oracles:

— $\mathcal{O}_{\mathsf{user}}((\mathcal{G}, \mathcal{P}_{\mathcal{G}}), (\mathsf{id}, \mathcal{S}_{\mathsf{id},\mathcal{G}}))$: This oracle emulates a user belonging to group $\mathcal{G}$ (with public value $\mathcal{P}_{\mathcal{G}}$), with identity id and secret credential $\mathcal{S}_{\mathsf{id},\mathcal{G}}$, and thus may consist of multiple interactions.

— $\mathcal{O}_{\mathsf{handshake}}(\mathcal{O}_{\mathsf{user}}((\mathcal{G}_1, \mathcal{P}_{\mathcal{G}_1}), (\mathsf{id}_1, \mathcal{S}_{\mathsf{id}_1,\mathcal{G}_1})), \mathcal{O}_{\mathsf{user}}((\mathcal{G}_2, \mathcal{P}_{\mathcal{G}_2}), (\mathsf{id}_2, \mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2})))$: This oracle emulates a handshake protocol execution between users id_1 (with secret $\mathcal{S}_{\mathsf{id}_1,\mathcal{G}_1}$) and id_2 (with secret $\mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2}$), and gives back the transcript of the execution.

— $\mathcal{O}_{\mathsf{handshake}}(\mathcal{A}(\texttt{INIT} : \mathcal{P}_{\mathcal{G}_1}), \mathcal{O}_{\mathsf{user}}((\mathcal{G}_2, \mathcal{P}_{\mathcal{G}_2}), (\mathsf{id}_2, \mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2})))$: This oracle emulates a handshake protocol execution between either the adversary or a user belonging to $\mathcal{G}_1$ emulated by the oracle $\mathcal{O}_{\mathsf{user}}$, and a user id_2 (with secret $\mathcal{S}_{\mathsf{id}_2,\mathcal{G}_2}$). In the first case, the adversary gets back the key computed by the protocol; In the second case, she gets back the transcript of the execution.

— $\mathcal{O}_{\mathsf{valid}}((\mathcal{G}, \mathcal{S}_{\mathcal{G}}), (K, \mathsf{transcript}))$: This oracle returns 1 whether $\mathsf{transcript}$ is a valid protocol execution transcript leading to a shared key K and 0 otherwise.

We can now describe the security games, presented in Figure 5.1 and parameterized by the bit b.

— *Impersonator resistance*: The adversary chooses the group $\mathcal{G}$ she wishes to attack and asks the simulator to initiate n executions between pairs of users (the second one being honest), giving her n keys $(K_i)_{i=1..n}$ for the first user. She then has to output another key K_{n+1} along with the corresponding transcript $\mathsf{transcript}$. She wins the game if the key K_{n+1} is valid, *i.e.* corresponds to a valid execution between two users belonging to the group $\mathcal{G}$. The success of the adversary is denoted as

$$\mathsf{Succ}^{\mathsf{impres}}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K}) = \Pr[\mathsf{Exp}^{\mathsf{impres}}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K}) = 1]$$
$$\mathsf{Succ}^{\mathsf{impres}}_{\mathcal{SH}}(\mathfrak{K}, t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{impres}}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K})$$

The scheme fulfills impersonator resistance if the success of any adversary running in time t is bounded by $q_e \times 2^{-m} \times \mathsf{Succ}^{\mathcal{L}}(t) + \varepsilon$, where q_e is the number of calls to $\mathsf{UserEnrollment}$, m is the min-entropy of the public values of the group $\mathcal{G}$, $\mathsf{Succ}^{\mathcal{L}}(t)$ is the maximal success an adversary can get in forging a valid secret value $\mathcal{S}_{\mathsf{id},\mathcal{G}}$ within time t and ε is a value negligible in the security parameter $\mathfrak{K}$.

— *Privacy*: The adversary chooses two groups $\mathcal{G}_0$ and $\mathcal{G}_1$ and two users id_1 and id_2 belonging to both groups. She then asks the simulator to initiate two executions (valid or not) between users id_1 and id_2, as members of group $\mathcal{G}_b$ or $\mathcal{G}_{1-b}$ and to

give her the transcript. She wins the game if he guesses correctly the bit b. The advantage of the adversary is defined as

$$\mathsf{Adv}^{\mathsf{privacy}}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K}) = \left|\Pr[\mathsf{Exp}^{\mathsf{privacy}-1}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K}) = 1] - \Pr[\mathsf{Exp}^{\mathsf{privacy}-0}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K}) = 1]\right|$$
$$\mathsf{Adv}^{\mathsf{privacy}}_{\mathcal{SH}}(\mathfrak{K}, t) = \max_{\mathcal{A}\leq t} \mathsf{Adv}^{\mathsf{privacy}}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K})$$

The scheme is private if the advantage of any adversary running in time t is bounded by a negligible value ε in the security parameter $\mathfrak{K}$.

— *Unlinkability*: The adversary chooses a group $\mathcal{G}$, knowing its secret value, and computes two secret values $\mathcal{S}_{\mathsf{id},\mathcal{G},0}$ and $\mathcal{S}_{\mathsf{id},\mathcal{G},1}$ for a user id of its choice. She asks the simulator to simulate two executions between her and this user for both secret values, and she has to guess the bit b (*i.e.* decide which secret value was used first). The advantage of the adversary is defined as

$$\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K}) = \left|\Pr[\mathsf{Exp}^{\mathsf{unlink}-1}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K}) = 1] - \Pr[\mathsf{Exp}^{\mathsf{unlink}-0}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K}) = 1]\right|$$
$$\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{SH}}(\mathfrak{K}, t) = \max_{\mathcal{A}\leq t} \mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{SH},\mathcal{A}}(\mathfrak{K})$$

The scheme is unlinkable if the advantage of any adversary running in time t is bounded by a negligible value ε in the security parameter $\mathfrak{K}$.

Definition 14 (Security for secret handshakes)**.** *A secret handshake protocol is claimed (t, ε)-secure if*

$$\mathsf{Succ}^{\mathsf{impres}}_{\mathcal{SH}}(\mathfrak{K}, t) + \mathsf{Adv}^{\mathsf{privacy}}_{\mathcal{SH}}(\mathfrak{K}, t) + \mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{SH}}(\mathfrak{K}, t) \leq q_e \times 2^{-m} \times \mathsf{Succ}^{\mathcal{L}}(t) + \varepsilon$$

where q_e is the number of calls to UserEnrollment*, m is the min-entropy of the public values of the group $\mathcal{G}$, $\mathsf{Succ}^{\mathcal{L}}(t)$ is the maximal success an adversary can get in forging a valid secret value $\mathcal{S}_{\mathsf{id},\mathcal{G}}$ within time t and ε is a value negligible in the security parameter $\mathfrak{K}$.*

5.4 Generic Construction

5.4.1 High-level Idea

The high-level idea of our round-optimal construction is very simple: The private value of a group $\mathcal{G}$ is a signing key $\mathcal{S}_{\mathcal{G}} = \mathsf{sk}_{\mathcal{G}}$, and its public value is the corresponding verification key $\mathcal{P}_{\mathcal{G}} = \mathsf{vk}_{\mathcal{G}}$. The proof of membership of a user id to the group $\mathcal{G}$ is a unique identifier signed under $\mathsf{sk}_{\mathcal{G}}$: $\mathcal{S}_{\mathsf{id},\mathcal{G}} = (m_{\mathsf{id}}, \sigma_{\mathsf{id}})$. Assume user i belonging to group $\mathcal{G}_i$ wants to execute the protocol with user j that he assumes belong to group $\mathcal{G}_j$. He computes an encryption C_i of his signature σ_i with random coins r_i, along with a private hash key hk_i and a projected hash key hp_i depending on $\mathcal{P}_{\mathcal{G}_j}$. He sends these three values to user j and receives the corresponding ones from him. His key is then the combination of a hash value (using hk_i, $\mathcal{P}_{\mathcal{G}_i}$, m_j and σ_j) and a projected hash value (using hp_j and r_i). The properties of the smooth projective hash function used (see below) ensures that his key is equal to that of user j as soon as everything has been computed honestly and the users have guessed the good group for each other.

The construction. We generically construct our Secret Handshake using a signature scheme $\mathcal{S}$, an asymmetric encryption scheme $\mathcal{E}$ and a smooth projective hash function SPHF. We describe each algorithm defined in Section 5.3.

Setup

The parameters param consist of the union of the parameters output by $\mathsf{Setup}_{\mathcal{S}}$, $\mathsf{Setup}_{\mathcal{E}}$ and $\mathsf{Setup}_{\mathsf{SPHF}}$. We assume the existence of a common reference string $\mathsf{crs} = \mathsf{ek}$ where $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}_{\mathcal{E}}$.

Create Group

$\mathsf{CreateGroup}(\mathcal{G})$ outputs a pair $\mathcal{S}_{\mathcal{G}} = \mathsf{sk}_{\mathcal{G}}$ and $\mathcal{P}_{\mathcal{G}} = \mathsf{vk}_{\mathcal{G}}$ where $(\mathsf{sk}_{\mathcal{G}}, \mathsf{vk}_{\mathcal{G}}) \leftarrow \mathsf{KeyGen}_{\mathcal{S}}$.

User Enrollment

$\mathsf{UserEnrollment}(\mathsf{id}, \mathcal{G}, \mathcal{S}_{\mathcal{G}})$ chooses a unique identifier m_{id} for user id, computes $\sigma_{\mathsf{id}} \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathcal{G}}, m_{\mathsf{id}}; \mu)$ for some randomness μ and outputs $\mathcal{S}_{\mathsf{id},\mathcal{G}} = (m_{\mathsf{id}}, \sigma_{\mathsf{id}}, \mathsf{vk}_{\mathcal{G}})$ to user id.

Handshake

$\mathsf{Handshake}((i, \mathcal{S}_{i,\mathcal{G}_i}), (j, \mathcal{S}_{j,\mathcal{G}_j}))$ is described in Figure 5.2 where $\mathcal{S}_{\mathcal{G}_k} = \mathsf{sk}_k$ and $\mathcal{P}_{\mathcal{G}_k} = \mathsf{vk}_k$ for $k \in \{i, j\}$. The public values are shared in a common reference string (CRS) between the users. The optional pub_i, pub_j variables represent some public parameters that can be sent, in clear, by any party.[2] Note that we need only one round of communication between both users thanks to the ProjKG algorithm which does not depend on C_i and C_j. The correctness is easily verified thanks to the properties of the SPHF since

$$\mathsf{Hash}(\mathsf{hk}_i, \mathcal{C}_j, m_j, \mathsf{vk}_j) = \mathsf{ProjHash}(\mathsf{hp}_i; r_j)$$
$$\mathsf{Hash}(\mathsf{hk}_j, \mathcal{C}_i, m_i, \mathsf{vk}_i) = \mathsf{ProjHash}(\mathsf{hp}_j; r_i)$$

$\underline{P_i}$: σ_i	$\mathsf{crs} : \mathsf{ek}$	$\underline{P_j}$: σ_j
$C_i \leftarrow \mathsf{Enc}(\mathsf{ek}, \sigma_i; r_i)$		$C_j \leftarrow \mathsf{Enc}(\mathsf{ek}, \sigma_j; r_j)$
$\mathsf{hk}_i \leftarrow \mathsf{HashKG}(1^{\mathfrak{K}})$		$\mathsf{hk}_j \leftarrow \mathsf{HashKG}(1^{\mathfrak{K}})$
$\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathsf{ek})$		$\mathsf{hp}_j \leftarrow \mathsf{ProjKG}(\mathsf{hk}_j, \mathsf{ek})$
	$\mathcal{C}_i, \mathsf{hp}_i, \mathsf{pub}_i \longrightarrow$	
	$\longleftarrow \mathcal{C}_j, \mathsf{hp}_j, \mathsf{pub}_j$	
$K_i = \mathsf{Hash}(\mathsf{hk}_i, \mathcal{C}_j, m_j, \mathsf{vk}_j) \cdot \mathsf{ProjHash}(\mathsf{hp}_j; r_i)$		$K_j = \mathsf{Hash}(\mathsf{hk}_j, \mathcal{C}_i, m_i, \mathsf{vk}_i) \cdot \mathsf{ProjHash}(\mathsf{hp}_i; r_j)$

Figure 5.2 – The $\mathsf{Handshake}$ algorithm

Theorem 3. *Under the* IND-CCA *property of the underlying encryption, the* $\mathsf{EUF} - \mathsf{CMA}$ *property of the signature and the smoothness of the smooth projective hash function, the*

2. In the instanciation of Section **??**, both parties send publicly parts of the signature for efficiency purpose although it removes the *unlinkable* property of the scheme.

protocol presented in Figure 5.2 is a secure secret handshake (in the sense of Definition 14).

Proof. The proof presented in this section follows in spirit the ones from [KV11,BBC+13a]. We are going to proceed in a sequence of games. We start from the real game, that we progressively modify to remove all but the trivial (guessing) attack. Users progressively stop knowing/using the secret values.

As described in the security model, the adversary will be able to run $\mathsf{Handshake}$ executions with another user (obtaining the key), or simulated executions via the $\mathcal{O}_{\mathsf{handshake}}$ oracle between two users (only obtaining the transcript).

— Game $\boldsymbol{G}_0$: This is the real game. $\mathsf{Adv}_0(\mathcal{A}) = \chi$.

— Game $\boldsymbol{G}_1$: We first modify the way we simulate the $\mathcal{O}_{\mathsf{handshake}}$ oracle between two honest users. We pick a random tuple σ of the correct size but that is not a valid signature, and replace C by and encryption of this σ. As both hashing keys are known, we set the shared key $K = K' = \mathsf{Hash}(\mathsf{hk}_i, \mathcal{C}_j, m_j, \mathsf{vk}_i) \cdot \mathsf{Hash}(\mathsf{hk}_j, \mathcal{C}_i, m_i, \mathsf{vk}_j)$.
Under the $\mathsf{IND\text{-}CPA}$ property (implied by the $\mathsf{IND\text{-}CCA}$ property) of the underlying encryption scheme, this is indistinguishable from $\boldsymbol{G}_0$. Hence
$$\mathsf{Adv}_1(\mathcal{A}) \leq \chi + q_E \cdot \mathsf{Adv}^{\mathsf{CPA}}(\mathfrak{K}) \leq \chi + q_E \cdot \mathsf{Adv}^{\mathsf{CCA}}(\mathfrak{K})$$

— Game $\boldsymbol{G}_2$: We modify again the way we simulate the $\mathcal{O}_{\mathsf{handshake}}$ oracle between two honest users. We now set a random value for the shared key. As the encrypted signature is invalid the encryption is outside the language. Under the smoothness of the underlying SPHF this is indistinguishable to the previous game. As smoothness relies on an information theoretic argument, we have $\mathsf{Adv}_2(\mathcal{A}) = \mathsf{Adv}_1(\mathcal{A}) + \mathsf{negl}(\mathfrak{K})$.

— Game $\boldsymbol{G}_3$: We now modify the way we simulate the $\mathcal{O}_{\mathsf{handshake}}$ oracle between the adversary and an honest user (initiated with a signature σ sent by the adversary), by accessing the decryption of the received words. When receiving a message (hp, C), we either encounter:

  — A replay of an existing flow: In this case, we know the hashing keys, and proceed as before

  — A new flow:

    — The decrypted signature is invalid: In this case, we can use a random value for the shared keys.

    — The decrypted signature is valid: The adversary managed to forge a valid signature, we declare she succeeds and terminate the game.

  The very last case is the only one increasing the advantage of the adversary by the advantage against the unforgeability of the signature, the second increases a very negligible advantage like in the previous game, and the first one does not change anything. Hence $\mathsf{Adv}_3(\mathcal{A}) = \mathsf{Adv}_2(\mathcal{A}) + \mathsf{Adv}^{\mathsf{uf}}(\mathfrak{K}) + \mathsf{negl}(\mathfrak{K})$.

— Game $\boldsymbol{G}_4$: We revisit again the way we simulate the $\mathcal{O}_{\mathsf{handshake}}$ oracle between the adversary and an honest user (initiated with a signature σ sent by the adversary).

If a message hp, C corresponds to partnered users $\mathsf{id}_{U'}, \mathsf{id}_U$, then we set the shared key for $\mathsf{id}_{U'}$ to be the same as the one for id_U, otherwise, we pick it at random. Under the smoothness, this is indistinguishable from the previous game, and so $\mathsf{Adv}_4(\mathcal{A}) = \mathsf{Adv}_3(\mathcal{A}) + \mathsf{negl}(\mathfrak{K})$.

— Game $\boldsymbol{G}_5$: We still need to alter the way we simulate the initialisation of the $\mathcal{O}_{\mathsf{handshake}}$ oracle: we are going to proceed as in $\boldsymbol{G}_1$ and encrypt a random tuple instead of a valid signature. As those values are then going to be decrypted by the second step of the simulation of the $\mathcal{O}_{\mathsf{handshake}}$ oracle (Games $\boldsymbol{G}_3$ and $\boldsymbol{G}_4$), we need to rely on the IND-CCA property of the underlying encryption. Hence $\mathsf{Adv}_5(\mathcal{A}) = \mathsf{Adv}_4(\mathcal{A}) + q_0 \cdot \mathsf{Adv}^{\mathsf{CCA}}(\mathfrak{K})$.

If one combines all those previous games:

$$\mathsf{Adv}_5(\mathcal{A}) = \mathsf{Adv}_0(\mathcal{A}) + \mathsf{Adv}^{\mathsf{uf}}(\mathfrak{K}) + (q_E + q_0) \cdot \mathsf{Adv}^{\mathsf{CCA}}(\mathfrak{K}) + \mathsf{negl}(\mathfrak{K})$$

One can then note, that in the final game, all the values used by the honest players are random, and not valid solutions. Hence, the only possibility for an adversary is to be able to forge a signature valid under an unknown verification key. Hence, $\mathsf{Adv}_5(\mathcal{A}) \leq \mathsf{negl}(\mathfrak{K}) + 2^{-m} \cdot \mathsf{Adv}^{\mathsf{uf}}(\mathfrak{K})$, which leads to

$$\mathsf{Adv}_0(\mathcal{A}) \leq (1 + 2^{-m}) \cdot \mathsf{Adv}^{\mathsf{uf}}(\mathfrak{K}) + (q_E + q_0) \cdot \mathsf{Adv}^{\mathsf{CCA}}(\mathfrak{K}) + \mathsf{negl}(\mathfrak{K})$$

□

5.5 Simple Construction in the ROM

5.5.1 SPHF on Cramer-Shoup commitments of BLS signature

In order to check a Cramer-Shoup ciphertext of a BLS signature with an SPHF, we use the framework of [BPV12b,BBC+13a] by adapting the signature (BLS here) and the SPHF used (the KV-SPHF on Cramer-Shoup was defined in [BBC+13b]).

— HashKG outputs $\mathsf{hk} = (\alpha, \beta, \lambda, \mu, \theta) \xleftarrow{\$} \mathbb{Z}_p^5$;

— $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{ek})$ outputs $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) = [\alpha g_1 + \beta g_2 + \lambda h + \mu c, \theta g_1 + \mu d]_1$;

— $\mathsf{Hash}(\mathsf{hk}, C, m, \mathsf{vk})$ computes $[(\alpha + \theta\chi)u_1 + \beta u_2 + \lambda e + \mu v]_T - [\mathcal{H}(m) \cdot \lambda \mathsf{vk}]_T$;

— $\mathsf{ProjHash}(\mathsf{hp}, r)$ computes $[r(\mathsf{hp}_1 + \chi \mathsf{hp}_2)]_T$ where r is the random element used in the Cramer-Shoup commitment;

Secret Handshake in the ROM. Applying the framework presented in Section 5.4 directly leads to the following instantiation. We assume the existence of a crs containing a Cramer Shoup encryption key, and appropriate hash functions: $\mathsf{crs} = \mathsf{ek} = ([g_1, g_2, c, d, h]_1, H_k, \mathcal{H})$, with $\mathsf{dk} = (x_1, x_2, y_1, y_2, z)$.

We instantiate our Secret Handshake protocol in the ROM with a BLS signature and the Cramer-Shoup encryption scheme. In part 5.5.1 we gave the SPHF that hashes words belonging to the language of ciphertexts of BLS signatures.

Thus, a direct application of our generic construction gives us the following algorithms.

Create Group

Any authority who wishes to create a group generates a BLS key pair $(\mathsf{vk}, \mathsf{sk}) = ([x]_2, x)$ where $x \xleftarrow{\$} \mathbb{Z}_p$.

User Enrollment

To enroll a user, a group administrator GA must sign his unique identifier id using its signing key sk. The user P receives $\sigma = [x\mathcal{H}(\mathsf{id})]_1$ and vk.

Handshake

We show in Figure 5.3 the whole handshake. For the sake of clarity, let GA_i be the group administrator of P_i (resp. GA_j for P_j). It has BLS key pair $(\mathsf{vk}_i, \mathsf{sk}_i) = ([x]_2, x)$ (resp. $(\mathsf{vk}_j, \mathsf{sk}_j) = ([y]_2, y)$) and Cramer-Shoup encryption key $\mathsf{ek} = (g_1, g_2, c, d, h, H_k)$.

Correctness can be easily verified. The keys K_i equals K_j computed by the users are equals if $\mathsf{vk}_i = \mathsf{vk}_j$.

The communication complexity is as follows: each party send a CS ciphertext (4 elements) and an SPHF projected key (2 elements).

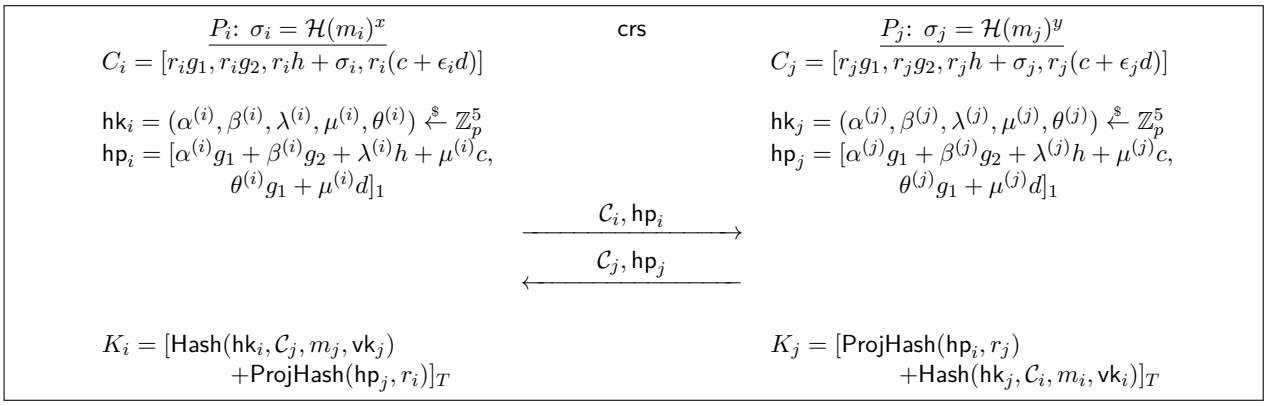

Figure 5.3 – The $\mathsf{Handshake}$ algorithm with CS ciphertexts of BLS signatures

5.6 Simple Construction in the Standard Model

We now present the main instantiation of this paper. We propose a secret handshake protocol that allows to sign group elements (and not only bitstrings), and that can be proven in the standard model. For that, we replace the inner signature by a signature derived from a Structure-Preserving signature defined in [KPW15].

We present the most efficient version of the scheme, which is not *unlinkable*. To make it unlinkable, at the cost of efficiency, we simply have to encrypt all elements of the signature. Otherwise, the signature elements given in clear would leak some

information on the party that executed the protocol already if the same signature is reused in another protocol.

Doing so increases both parties' efficiency in the Secret Handshake protocol but if the same signature is reused in another protocol, those randomized elements will leak one information: the same party that executed the protocol already. Depending if unlinkability is expected, one may want to encrypt every single element.

5.6.1 SPHF on Cramer-Shoup commitments of Structure-Preserving signatures

We use the same framework as in 5.4, but we only have to encrypt the two group elements of σ_1 to avoid leaking the signature: C_i denotes the encryption of $\sigma_{1,i}$, $C_i \stackrel{\text{def}}{=} [r_i g_1, r_i g_2, r_i h + \sigma_{1,i}, r_i(c + \chi_i d)]_1$.

- HashKG outputs $\mathsf{hk} = \{(\alpha_i, \beta_i, \mu_i, \theta_i)\}_{i\in\{1,2\}} \cup \{\lambda\} \xleftarrow{\$} \mathbb{Z}_p^9$;
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{ek})$ outputs $\mathsf{hp} = \{([\alpha_i g_1 + \beta_i g_2 + \lambda h + \mu_i c, \theta_i g_1 + \mu_i d]_1)\}_{i\in\{1,2\}}$;
- $\mathsf{Hash}(\mathsf{hk}, C, m, \mathsf{vk})$ computes $\big(\sum_{i\in\{1,2\}}[\alpha_i u_{1,i} + \beta_i u_{2,i} + \lambda_i e_i + \mu_i v_i + \theta_i \chi_i u_{1,i} + a_i]_T\big) - [\lambda((1, m)\mathbf{C} + \sigma_2(\mathbf{C_0} + \tau\mathbf{C_1}))]_T$;
- $\mathsf{ProjHash}(\mathsf{hp}, r_1, r_2)$ computes $\sum_{i\in\{1,2\}}[r_i(\mathsf{hp}_{1,i} + \chi_i \mathsf{hp}_{2,i})]_T$ where r_i are the random elements used in the Cramer-Shoup commitments;

Secret Handshake

Applying the framework presented in Section 5.4 directly leads to the following instantiation. We assume the existence of a crs containing the encryption key ek for a Cramer-Shoup encryption.

Create Group

Any authority who wishes to create a group generates a signature key pair

$$(\mathsf{sk}, \mathsf{vk}) = ((\mathbf{K}, [\mathbf{P_0}]_1, [\mathbf{P_1}]_1, [\mathbf{B}]_1), ([\mathbf{C_0}]_2, [\mathbf{C_1}]_2, [\mathbf{C}]_2, [\mathbf{A}]_2)).$$

User Enrollment

To enroll a user, a group administrator must sign his unique identifier using its signing key sk. The user P receives $\sigma = (\sigma_1, \sigma_2, \tau)$, the verification key vk.

Handshake

The protocol is described in Figure 5.4.

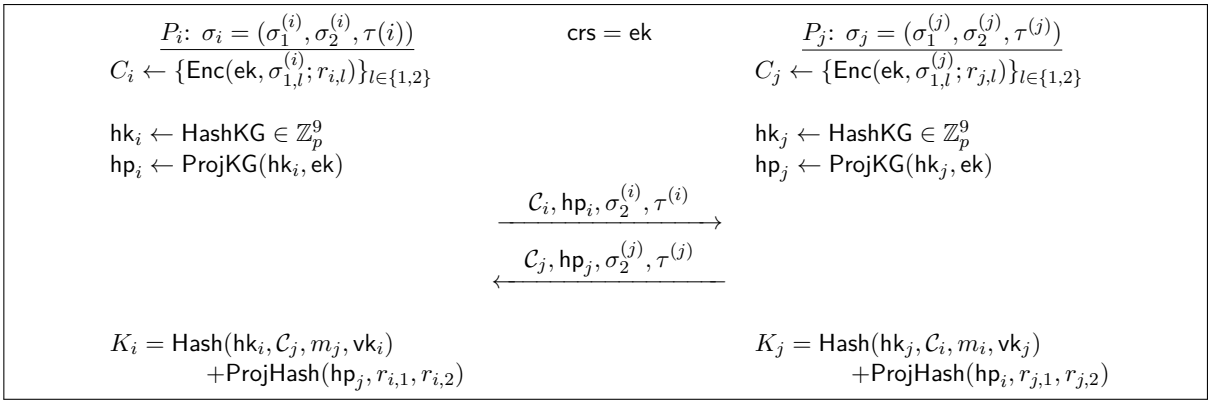

Figure 5.4 – The Handshake algorithm with CS ciphertexts of SPS signatures

5.7 Implementation/Applications

In this section we present and evaluate an implementation of both schemes. We also demonstrate the practicability of our secret handshake by showing how it could be integrated into real-world network protocols such as TLS or SSH.

5.7.1 Performance of our Schemes

In order to give a proof-of-concept of our schemes, we implemented all the components in Python 3, using the Charm library [AGM+13], a framework allowing to quickly prototype cryptographic schemes. We chose to use the already implemented *pairinggroup* class to manipulate elliptic curves and used the few available built-in pairing-friendly elliptic curves: MNT159, MNT201, MNT224 and SS512, which respectively provide security levels of 70, 90, 100 and 80 bits. Products of matrices over $\mathbb{Z}_p$ are done with a non-optimized algorithm. We conducted our experiments on a Ubuntu virtual machine with 4 GB of RAM and an Intel Core i5 processor running at 2.3 GHz on the host processor.

Even though our implementation was not designed with optimization in mind, Table 5.1 show that the obtained execution times are quite practical. We executed the protocol fifty times and took the average value for each step. Regarding communication costs, our schemes are very efficient: few elements are exchanged, the protocol only requires a message from each party executing the handshake, and both messages can be sent concurrently.

A brief complexity comparison of the existing Secret Handshake schemes with ours can be found in Table 5.2. It summarizes our contribution, showing we achieve a similar or better level of security in a single concurrent round at the cost of a reasonable computational overhead. We denote by P, M, E the classical pairing, multiplication and exponentiation operations. Some algorithms also use oblivious encryption that we denote here with Enc/Dec.

BLS+CS

Curve	MNT159	MNT201	MNT224	SS512
AuthSetup + Create Group	68 ms	98 ms	119 ms	16 ms
User Enrollment	< 1 ms	1 ms	1 ms	4 ms
Handshake	41 ms	53 ms	64 ms	39 ms

SPS+CS

Curve	MNT159	MNT201	MNT224
AuthSetup + Create Group	167 ms	241 ms	273 ms
User Enrollment	2 ms	3 ms	4 ms
Handshake	147 ms	190 ms	235 ms

Table 5.1 – Numerical evaluation

Scheme	**Cost**	**# Rounds**	***Unlinkability***	**Security**
[BDS+03b]	1P	3	no	ROM
[CJT04]	1Enc + 1Dec	3	no	ROM
[XY04]	$\mathcal{O}(k)$Enc	3	k-limited	ROM
[AKB07]	2P + 2E	3	yes	BDH
[GX11]	1P + 1M	3	yes	ROM
BLS + CS	3P + 19E	1	yes	ROM (CDH)
SPS + CS	10P + 50E	1	no	SXDH
SPS + CS	14P + 142E	1	yes	SXDH

Table 5.2 – Comparison of performance of existing Handshake algorithms.

5.7.2 Integration with TLS

An important property of our secret handshake schemes is that they are round-optimal, in the sense that both users only need to send one flow of data, and that the flows can be sent concurrently (*i.e.* without waiting for the other to be delivered). Our schemes can thus be easily integrated in network protocols such as TLS or SSH, while allowing for strong security properties:

— privacy and anonymity: A user only needs to reveal their membership to a given organization to authenticate to a server, not their identity.

— security with respect to the authority: Thanks to the privacy property, no observer can infer the result of an interaction (success or failure), and more importantly, which credential the client used to authenticate.

In standard TLS connections, the computational overload (one key exchange plus two signatures and verifications) is negligible in comparison to the communication cost of the session establishment. For example, if we consider symmetric curves offering a security of 128 bits, such as NISTP256, OpenSSL requires between 0.2 and 3 ms for ECDSA and ECDH operations, which leads to less than 15 ms in total. The network cost, measured in RTTs (Round-Trip Times, which roughly correspond to 100 ms in typical situations), is 1 or 2 RTT with TLS.

For our schemes, the computational cost is higher (even if only the handshake step

must be taken into account). One of the reasons is that we need to compute pairings (on the far less efficient asymmetric curves for SPS+CS), whereas pairings are not used in TLS (which allows to use more efficient curves). This would be the case even with a more optimized implementation.

However, even with our current implementation, the computation cost adds between 0.5 to 2.5 RTTs (depending on the version and the curve), which leads to a 1.5 to 4.5-RTT long establishment time, which seems reasonable compared to regular TLS (1 to 2 RTTs), making them practical for a use for servers needing the strong additional security properties they offer. This is even more true if we consider SSH, which requires 4 to 5 RTTs to initiate a typical connection.

Chapter 6

Code-based Round-Optimal Oblivious Transfer

The concept of Oblivious Transfer (OT) was introduced by Rabin in 1981 [Rab81]. In the simplest form of the protocol, the 1-out-of-2 OT, a sender possesses two messages m_0, m_1 and a receiver chooses a bit b. After a successful execution of the protocol, the receiver obtains message m_b while learning nothing about m_{1-b} and the sender does not know which of the two messages has been requested.

Contents

6.1 General overview

Such a scheme has been shown to be *complete* [Kil88,IPS08] in the sense that secure Multi-Party Computation (MPC) can be obtained directly from it.

6.1.1 State of the Art

The work of [CLOS02] motivated the use of the Universal-Composability (UC) framework of [Can01] when constructing OT schemes. Indeed, proving a scheme secure in the UC model guarantees that composing it in a larger protocol will preserve security. As OT is regarded as a building block for MPC and as such is ran under composition, it is of importance to have it in the UC model. Another important notion that divides many works in the area of OT is the adversary corruption model which can be *static* or *adaptive*. In the former model, corrupted parties are decided at the start of the protocol while adaptive security is a stronger model in which the adversary can corrupt honest parties at any time during the protocol. It captures better real-world scenarios where corruption can happen at anytime e.g. if an adversary acquires the control of a computer during execution of the protocol.

Many OT instantiations have been proposed to reach greater efficiency in terms of round, computation or communication costs [NP01, NP05, HK12, CO15]. In [HK07], the authors achieved the first round-optimal OT while [PVW08] propose a general framework for round-optimal UC-secure OT, both are in the static setting. In the adaptive security model, [BC16,BCG17,CKWZ13,GWZ09,BDD$^+$17] proposals lack efficiency and round-optimality.

In a recent work from [BPRS17], the authors present the first round-optimal adaptively UC-secure construction of OT relying on the Decisional Diffie-Hellman (DDH) problem. Their approach is closely related to that of [PVW08] but it does not seem to be instantiable using post-quantum hardness assumption. Our work follows up on this in the sense that our framework can be instantiated, as we show, from Rank Metric Codes-based cryptographic schemes which are believed to be quantum-computer resistant .

Since the original paper [Rab81], several instantiations of OT protocols have appeared in the literature, including proposals in the UC framework. Some instantiations tried to achieve low communication costs [PVW08], while others like Choi et al. [CKWZ13,BC15] proposed a generic method to achieve Oblivious Transfer with adaptive corruptions. However, among all the existing articles, only the schemes from [PVW08] (an ad-hoc construction based on lattices) and [BC15] (a generic construction relying on [KV09]), or [BCV19] (a variant using [BBDQ18]) are UC-Secure in a Post-Quantum setting, and they rely on lattice-based cryptography. Unfortunately, the first construction only fulfills static security. The other one offers adaptive security, but relies on standard-model lattice-based SPHF constructions with costly decryption procedure (either with repetitions or a super polynomial modulus)

Some code-based Oblivious Transfers exist like [KMO08, BDD$^+$17], however none of them managed to be proven in the UC setting up to now.

6.1.2 Contributions:

— First, we revisit a framework for round-optimal OT already seen in the literature which requires a certain property on the public keys. Informally, given a random value $\mathcal{T}$, it should not be possible to have two public keys of an asymmetric encryption scheme $\mathsf{pk}_0, \mathsf{pk}_1$ satisfying the relation $\mathcal{T} = \mathsf{pk}_0 \star \mathsf{pk}_1$. To prove this framework adaptively UC-secure, we revise the classical ideal functionality of OT.

— Secondly, we propose multiple instantiations of this framework using rank metric code-based encryption scheme (NIST competitors) such as HQC [AAB+17b] and RQC [AAB+17a] with performance evaluation.

We thus propose the first construction of adaptively UC-secure round-optimal OT using quantum-resistant hardness assumptions.

6.2 Framework

We now introduce a framework to build Oblivious Transfer protocols in the Random Oracle Model.

6.2.1 Star product

To describe our framework, we need to use a Public-Key Encryption scheme whose keys have the following properties:

Definition 15 (Star product)**.** *Let $\mathcal{T}$ be a random value, $(\mathsf{pk}_1, \mathsf{pk}_2)$ be two values lying in the public key space of the considered PKE, and $\star$ be on operation between two elements of the public key space.*

If $\mathsf{pk}_1 \star \mathsf{pk}_2 = \mathcal{T}$, we want the following properties :

— ***Sender security:*** *The Receiver must not be able to know the secret keys associated to both of the public keys*

— ***Receiver security:*** *The Sender must not be able to tell which secret key is know to the receiver*

6.2.2 Framework

In figure 6.1, we present a framework from the earliest version of [BDD+17], that we adapt (in particular we add erasures at the critical steps [1]) to achieve **Practical** Oblivious-Transfer. We present later a new UC functionality in Figure 6.2 and show this framework can achieve it.

The classical functionality for Oblivious Transfer has been achieved in many cases, however it often artificially complicates the construction.

1. Without them it would not be possible to prove the framework within our functionality, in particular in case of the server corruption after its flow.

Assuming a $\mathsf{crs} = \mathcal{H}_1, \mathcal{H}_2$
Index query on b:
Receiver will generate two encryption keys knowing only one secret key sk_b

— Picks a random seed t, set $\mathcal{T} = \mathcal{H}_1(t, \mathsf{sid}, \mathsf{ssid})$
— For chosen b, run $\mathsf{pk}_b = \mathsf{KeyGen}(1^k, \mathsf{crs})$
— If $b = 1$, sets $\mathsf{pk}_0 = \mathcal{T} \oplus \mathsf{pk}_1$
— Sends (t, pk_0), keeps b, sk_b and erases the rest

Database input (m_0, m_1):
Sender will encrypt both messages m_0, m_1 under their respective keys

— Upon receiving (t, pk_0)
— Regenerates $\mathsf{pk}_1 = \mathsf{pk}_0 \oplus \mathcal{T}$ with t, pk_0
— For each $b \in \{0, 1\}$
  — Picks a random $K_b \in \{0,1\}^\lambda$, queries $K'_b = \mathcal{H}_2(K_b, \mathsf{sid})$
  — Compute $C_b = \mathsf{Enc}(\mathsf{pk}_b, K_b; r_b)$
  — Computes $u_b = m_b \oplus K'_b$
— Sends (u_0, u_1, C_0, C_1), erases everything afterwards

Data recovery:
Receiver recovers the wanted line m_b

— Upon receiving (u_1, u_2, C_0, C_1)
— For the chosen b, computes $K_b = \mathsf{Decrypt}(\mathsf{sk}_b, C_b)$
— Recovers $m_b = u_b \oplus \mathcal{H}_2(K_b, \mathsf{sid})$

Figure 6.1 – A simple-OT achieving the Practical-OT Functionality

Inspired by the work of [BPRS17], we present an ideal functionality for Oblivious Transfer protocols in figure 6.2. Our framework which appears to be similar as in the earliest version of [BDD+17], as shown in [LM18], cannot be proven UC-secure according to this functionality due to a "timing bug".

In this paper, we propose a new functionality more adapted to the real world. At a high level, we propose to split the `Receive` element into two parts. An $(\texttt{Emit}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ which models that the server is seen sending a message, and a new $(\texttt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ which models the fact the user receives **and interprets** the flow.

6.3 New functionality

It should be noted that every protocol proven secure in the classical OT functionality is also secure in the one presented in Figure 6.2. In the other direction, every secure

The functionality $\mathcal{F}_{\text{(1,2)-Practical-OT}}$ is parametrized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

— **Upon receiving an input (Send, sid, ssid, P_i, P_j, (m_0, m_1)) from party P_i**, with $m_i \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_0, m_1))$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.

— **Upon receiving an input (Emit, sid, ssid, P_i, P_j) from party P_j,**: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{wait})$ and reveal $(\mathtt{Emit}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Emit-message with the same ssid from P_j.

— **Upon receiving an input (Receive, sid, ssid, P_i, P_j, s) from party P_j,** with $s \in \{0,1\}$: if a $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{wait})$ existed supersede it with the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$, and reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Receive-message with the same ssid from P_j.

— **Upon receiving a message (Sent, sid, ssid, P_i, P_j) from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_0, m_1))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to P_i and ignore further Sent-message with the same ssid from the adversary.

— **Upon receiving a message (Emitted, sid, ssid, P_i, P_j) from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_0, m_1))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{wait})$ is not recorded; otherwise send $(\mathtt{Emitted}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to P_j and ignore further Emitted-message with the same ssid from the adversary.

— **Upon receiving a message (Received, sid, ssid, P_i, P_j) from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_0, m_1))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, m_s)$ to P_j and ignore further Received-message with the same ssid from the adversary.

Figure 6.2 – Ideal Functionality for 1-out-of-2 Oblivious Transfer $\mathcal{F}_{\text{(1,2)-Practical-OT}}$

protocol within this functionality is also a secure weak-Oblivious Transfer [MR16] [2].

6.3.1 Security Proof in this new functionality

Theorem 4. *The framework explained in Figure 6.1 achieves the functionality presented in Figure 6.2 in the ROM, under the security of the star product.*

To prove this theorem, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$. We prove the adaptive version of the protocol. The proof of the static version can be obtained by removing the parts related to adaptive

2. This is trivial, as weak Oblivious Transfer do not require s in the Receive message, hence are encompassed by Emit.

version from the proof below. We denote as P_i the sender (the server) and P_j the receiver (the user).

Essentially, one first send random values for $\mathcal{T}$, and equivocate using the ROM if need be, then, under the IND-CPA property of the Encryption scheme, one can send dummy values for both u_i. Finally at the time of decryption, one can program the ROM so that when queried a hash on K_b, one can deduce the knowledge of sk_b and so decide that the queried $s = b$, and use the ROM to lead to m_s. This allows to simulate the Send, Emit, Receive-queries respectively to the ideal functionality.

More details follow:

Game G_0: This is the real game.

Game G_1: In this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs (m_0, m_1) and s sent by the environment to the sender and the receiver. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game G_2: We first deal with **honest senders** P_i: when receiving a commitment t, pk_0, the simulator, instead of computing the mask u_b, for $b = 0, 1$ with the key K_b, it chooses u_b at random.

When queried on $\mathcal{H}_2(K_b, \mathsf{sid})$, the ROM returns $u_b \oplus m_b$. This game is indistinguishable from the previous one.

In case of corruption, everything has been erased. This game is thus indistinguishable from the previous one under the ROM.

Game G_3: We now deal with **honest receivers** P_j: we replace all the commitments t, pk_0 in Step 1 of the index query phase of honest receivers by honestly generating pk_0 and pk_1 knowing their respective secret keys $\mathsf{sk}_0, \mathsf{sk}_1$, and programming the ROM so that $\mathcal{T} = \mathsf{pk}_0 \star \mathsf{pk}_1$. We then store $(\mathsf{sk}_0, \mathsf{sk}_1)$.

Under the difficulty of distinguishing an honestly generated $\mathcal{T}$ from a crafted one, this game is similar to the precedent.

In case of corruptions, one reveals the valid sk_b and erases the other one.

Game G_4: We now deal with **honest receivers P_j, receiving**: we simply recover m_b by using the appropriate sk_b as usual.

Game G_5: We can now make use of the functionality, which leads to the following simulator:

— when receiving a Send-message from the ideal functionality, which means that an honest receiver has sent a first flow, the simulator generates secret keys, and programs the ROM so that $\mathcal{T}$ is the star product of the associated public keys, and sends the corresponding flow;

— after receiving the said flow (from an honest or a corrupted sender) and a Emitted-message from the ideal functionality, which means that an honest receiver has sent an index query, the simulator sends honest C_b and random u_b, and uses it to send the corresponding Emit-message to the ideal functionality;

— when receiving (u_0, u_1, C_0, C_1) from the adversary (a corrupted sender), the simulator uses both $\mathsf{sk}_0, \mathsf{sk}_1$ to recover u_0, u_1. It uses them to send a `Receive`-message to the ideal functionality.

— when receiving a `Received`-message from the ideal functionality, together with m_s, on behalf of a corrupted receiver, from the extracted s, instead of proceeding as the sender would do on (m_0, m_1), the simulator proceeds on answering $u_s \oplus m_s$ when queried on $\mathcal{H}_2(K_b, \mathsf{sid})$;

— when receiving a flow, generated by an honest sender (by the simulator itself), the simulator proceeds by answering randomly for every $\mathcal{H}_2$ queries

Any corruption before the end does not alter the simulation, a corruption at the `Receive` phase reveals s, and alters one answer of the random oracle... In all cases, a corruption is undetectable as all traces of simulations are erased.

Remark:

It should be noted, that this split in the functionality, avoids the *trick* used in various protocols [GWZ09, ABB+13, BC15], where they need an artificial first flow, to encrypt a hiding mask.

6.4 Instantiations

In this section we propose concrete instantiations of the framework presented in Section 6.2 using the code-based cryptosystems HQC [AAB+17b] (Hamming metric) and RQC [AAB+17a] (rank metric). We start by defining the star product of Section 6.2.1 for the chosen cryptosystems and then discuss the possible bandwidth/complexity tradeoffs.

6.4.1 High-level details on the security when instantiated with HQC or RQC

In HQC and RQC, a public key pk consists of two vectors $(\boldsymbol{h}, \boldsymbol{s}) \in \mathbb{F}^{2n}$, such that $\boldsymbol{s} = \boldsymbol{x} + \boldsymbol{hy}$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are of weight w. Thus we can write $\mathcal{T}$ as $(\mathcal{T}_1, \mathcal{T}_2)$, where $\mathcal{T}_i \in \mathbb{F}^n$ and $\mathsf{pk}_1 + \mathsf{pk}_2 = \mathcal{T} \Leftrightarrow (\boldsymbol{h}_1 + \boldsymbol{h}_2 = \mathcal{T}_1, \boldsymbol{s}_1 + \boldsymbol{s}_2 = \mathcal{T}_2)$.

To prove these instantiations of the framework are secure, we need to assume that $\boldsymbol{h}_i$ cannot be chosen by the receiver during the Keygen operation. This can be achieved by setting:

— $\boldsymbol{h}_1 = \mathcal{H}(\mathcal{T})$

— $\boldsymbol{h}_2 = \boldsymbol{h}_1 + \mathcal{T}_1$

When receiving the public keys, the sender checks that the $\boldsymbol{h}_i$ were computed legitimately.

To prove the security of our instantiations, we use two hard problems given in definitions 7 and 8 in chapter 2, section 2.2.

Remark:

The HQC and RQC cryptosystem use a cyclic structure to reprensent matrices in a more compact way, hence a matrix $\boldsymbol{H} \in \mathbb{F}^{(n-k)\times n}$ can be represented by an associated

vector $\boldsymbol{h} \in \mathbb{F}^n$. We refer the reader to [AAB+17b] and [AAB+17a] for more details about this.

Proposition 16. *Under the assumption that the Syndrome Decoding (respectively Rank Syndrome Decoding) problem [7] is hard, the + operation is a valid star product for the public keys of the HQC (respectively RQC) cryptosystem.*

Proof. **Sender security:**

Finding two public keys pk_1 and pk_2 such that $\mathsf{pk}_1 + \mathsf{pk}_2 = \mathcal{T}$, knowing that $\boldsymbol{h}_1 = \mathcal{H}_1(\mathcal{T})$ and $\boldsymbol{h}_2 = \boldsymbol{h}_1 + \mathcal{T}_1$, is equivalent to finding $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{s}_1, \boldsymbol{s}_2)$ such that :

— $\boldsymbol{s}_1 = \boldsymbol{x}_1 + \boldsymbol{h}_1\boldsymbol{y}_1$

— $\boldsymbol{s}_2 = \boldsymbol{x}_2 + \boldsymbol{h}_2\boldsymbol{y}_2$

— $\boldsymbol{s}_1 + \boldsymbol{s}_2 = \mathcal{T}_2$

Where the weight of $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ is w, with w a parameter of the cryptosystem. Solutions to this system are also solutions of the equation:

$$\mathcal{T}_2 = (\boldsymbol{x}_1 + \boldsymbol{x}_2) + \boldsymbol{h}_1\boldsymbol{y}_1 + \boldsymbol{h}_2\boldsymbol{y}_2$$

Which is an instance of the syndrome decoding problem 7 where:

— The parity check matrix matrix $\boldsymbol{H}$ is generated by $(1, \boldsymbol{h}_1, \boldsymbol{h}_2)$

— The error vector is $((\boldsymbol{x}_1 + \boldsymbol{x}_2), \boldsymbol{y}_1, \boldsymbol{y}_2)$

— The syndrome is $\mathcal{T}_2$

Hence we can reduce the problem of finding keys that break the sender security in our instantiation to solving these particular instances of syndrome decoding. Let $\mathcal{A}$ denote an adversary against these particular instances. We are going to show that $\mathcal{A}$ can be used to solve hard generic instances of syndrome decoding.

In the following, we are going to consider error vectors of length $3n$. The notation (w_1, w_2, w_3) means that the weight of a vector is w_1 if we take the first n coordinates, w_2 if we take the following n, and w_3 if we take the last n.

Hamming metric

We want to solve a random syndrome decoding instance $(\boldsymbol{H}, \boldsymbol{s}, 3w)$ where $\boldsymbol{H} \in \mathbb{F}^{n \times 3n}$ and $\boldsymbol{s} \in \mathbb{F}^n$.

First, by remarking that random instances of weight $3w$ and length $3n$ have, with good probability, a solution of the form (w, w, w), we can reduce the generic problem to these specific instances.

We then use $\mathcal{A}$ to build an adversary $\mathcal{A}'$ which, on input $(\boldsymbol{H}, \boldsymbol{s})$, proceeds as follows:

— Sample a random vector $\boldsymbol{e}' \in \mathbb{F}^3 n$ of weight $(w, 0, 0)$

— Send $(\boldsymbol{H}, \boldsymbol{s} + \boldsymbol{H}\boldsymbol{e}'^{\top})$ to $\mathcal{A}$. Then:

  — If $\mathcal{A}$ outputs a vector $\boldsymbol{e}''$ then output $\boldsymbol{e}'' - \boldsymbol{e}'$

  — Else output $\emptyset$

$\mathcal{A}$ returns error vectors that have their weight bounded by $4w$, which is below the Gilbert-Varshamov bound. This ensures the uniqueness of the solution, which means that $\boldsymbol{e}'' - \boldsymbol{e}'$ is indeed a solution to the instance $(\boldsymbol{H}, \boldsymbol{s})$.

Rank metric

We want to solve a random rank syndrome decoding instance $(\boldsymbol{H}, \boldsymbol{s}, 2w)$ where $\boldsymbol{H} \in \mathbb{F}_{q^m}^{n \times 2n}$ and $\boldsymbol{s} \in \mathbb{F}_{q^m}^{n}$.

We use $\mathcal{A}$ to build an adversary $\mathcal{A}'$ which, on input $(\boldsymbol{H}, \boldsymbol{s})$, proceeds as follows:

— Sample a random vector $\boldsymbol{h}_2 \in \mathbb{F}_{q^m}^{n}$

— Build $\boldsymbol{H}'$, the concatenation of $\boldsymbol{H}$ and the square matrix generated by the vector $\boldsymbol{h}_2$. This results in a $n \times 3n$ matrix over $\mathbb{F}_{q^m}$.

— Sample a random vector $\boldsymbol{e}'$ of weight $(w, 0, w)$

— Send $(\boldsymbol{H}', \boldsymbol{s} + \boldsymbol{H}'\boldsymbol{e}'^{\top})$ to $\mathcal{A}$. Then:

  — If $\mathcal{A}$ outputs a vector $\boldsymbol{e}''$ then output the first $2n$ coordinates of $\boldsymbol{e}'' - \boldsymbol{e}'$

  — Else output $\emptyset$

In the rank metric $\mathcal{A}$ returns error vectors that have their weight bounded by $2w$, which is again below the Gilbert-Varshamov bound. This ensures the uniqueness of the solution, which means that $\boldsymbol{e}'' - \boldsymbol{e}'$ is indeed a solution to the instance $(\boldsymbol{H}, \boldsymbol{s})$.

From that we deduce that finding $(\mathsf{pk}_1, \mathsf{sk}_1)$ and $(\mathsf{pk}_2, \mathsf{sk}_2)$ such that $\mathsf{pk}_1 + \mathsf{pk}_2 = \mathcal{T}$ reduces to finding a solution to an instance of the syndrome decoding problem.

Receiver security:
To distinguish between a legitimately generated public key and some $(\boldsymbol{h}, \boldsymbol{s})$ randomly sampled, the sender would have to solve an instance of the decisional version of the syndrome decoding problem, which is easily reduced to the search version. □

6.5 Implementation

6.5.1 Implementation details

Since the HQC and RQC schemes are implemented as KEM schemes instead of PKE schemes, we use the conversion 6.3 to encrypt 512-bit messages. To decrypt, we simply use the left part of C to recover the shared secret and XOR it with the right part of C to recover m.

1: **Input:** The 512-bit message to encrypt m, the secret key sk
2: **Output:** A ciphertext C of m
3: $(C_1, shared\_secret) \Leftarrow \mathsf{Encaps}(\mathsf{sk})$
4: $m' \Leftarrow m \oplus shared\_secret$
5: **return** $C = C_1 || m'$

Figure 6.3 – KEM to PKE conversion used in our implementation

6.5.2 Bandwidth and performances

Bandwidth:

Our implementation uses:

— 40 bytes to represent t and 64 bytes to represent sid, hence a 104 bytes overhead over transmitting the public key

— 128 bytes to represent (u_1, u_2) and 128 bytes of overhead coming from the KEM to PKE conversion, hence a 256 bytes overhead over transmitting two ciphertexts

Using that we can compute the bandwidth used by our instantiations for security levels 128, 192 and 256. The results are described table 6.1. For RQC, we chose to use the parameter RQC-III as a 128 bits of security parameter to take into account the recent improvments on algebraic attacks from [BBB+19].

Parameter	Security level	Receiver bandwidth	Sender bandwidth
hqc-128-1	128	6274	12724
hqc-192-1	192	11022	22218
hqc-256-1	256	16002	32176
RQC-III	128	2388	9360

Table 6.1 – Bandwidth (in bytes) used by our instantiations of the OT framework

Performances:

The most costly operations in the OT framework are by far the KeyGen, Enc and Decrypt operations, hence the performance of the Oblivious Transfer is really close to the performances of the HQC and RQC schemes. The timings are shown table 6.2 and are given in million of CPU cycles. The computations were performed using an Intel® Core™ i5-7440HQ.

Parameter	Receiver (step 1)	Sender	Receiver (step 2)
hqc-128-1	0.20	0.67	0.45
hqc-192-1	0.35	1.18	0.76
hqc-256-1	0.53	1.85	1.20
RQC-III	0.52	1.51	3.24

Table 6.2 – Timings (in millions of cycles) of our instantiations of the OT framework

Chapitre 7

Conclusion

Dans cette thèse, nous présentons donc trois résultats dont deux possédant une base relativement commune et l'autre appartenant à un domaine légèrement éloigné. Ainsi, les deux premiers articles nous donnent des schémas de *Blind Signature* ainsi qu'un protocole de *Secret Handshake*, aux variantes près, avec, au coeur de leur construction, une signature numérique préservant la structure des éléments. Le troisième, lui, décrit un schéma d'*Oblivious Transfer* dans le monde de la cryptographie post-quantique.

Dans le chapitre 4, nous nous inscrivons directement dans la continuité d'articles décrivant une nouvelle primitive, la *Signature on Randomizable Ciphertexts*, qui permet de signer des classes de chiffrés d'un même message puis d'extraire des signatures directement sur le message. Un des résultats de ce travail est le fait d'avoir construit cette SRC à partir d'une *Structure-Preserving Signature* particulièrement efficace et dont les instantiations peuvent être sous plusieurs hypothèses de difficulté, DLin et SXDH, dans le modèle standard. En appliquant alors le framework de Fischlin, nous obtenons ainsi, à l'aide de la méthodologie Groth-Sahai, une *Blind Signature* qui cette fois-ci réunie trois propriétés : elle est optimale en nombre de tours, sa sécurité est prouvée dans le modèle standard et le coût en communication ne dépend pas de la taille des messages signés.

Toujours grâce à cette même *Structure-Preserving Signature*, en la remaniant légèrement, nous obtenons dans le chapitre 5 un protocole de *Secret Handshake* lui aussi optimal en nombre de tours et efficace en pratique. Là aussi, notre travail s'inscrit dans la continuité d'une publication sur les *Language-Authenticated Key Exchange*, une généralisation de l'échange de clé authentifié par la satisfaction de relations algébriques définies pour le langage voulu. En effet, le protocole de *Secret Handshake* permet à deux utilisateurs de s'authentifier s'ils appartiennent à une même organisation tout en ne divulguant aucune information sur cette appartenance. Nous modélisons la preuve de l'appartenance à l'organisation secrète par la possession d'un chiffré sur un message. Ainsi, à l'aide d'une *Smooth Projective Hash Function* instantiée correctement sur le langage des chiffrés d'une signature sur un certain message par rapport à une certaine clé de vérification et par rapport à une certaine clé de chiffrement, nous arrivons à créer un secret commun pour les deux utilisateurs. Les propriétés de la SPHF font que cela n'arrive que si leurs preuves d'appartenance satisfont le même langage. Encore une fois,

l'utilisation de la SPS nous permet d'obtenir une efficacité que nous démontrons également par la pratique en implémentant les différentes variantes de notre protocole. Les temps d'exécutions sont peu supérieurs, par exemple en intégrant le SH dans TLS, à un protocole d'AKE classique.

Notre troisième publication traite de la primitive d'*Oblivious Transfer*, nécessaire dans les protocoles de *Multi-Party Computation*. Nous y présentons ainsi, dans le chapitre 6, une instantiation d'un framework pour obtenir un OT optimal en nombre de tours reposant sur un algorithme de chiffrement asymétrique possédant une propriété particulière sur les clés publiques utilisées. La première étape de notre résultat est de revisiter la fonctionnalité classique de l'OT dans le modèle de l'*Universal Composability* afin d'obtenir une preuve de sécurité pour le framework utilisé dans le modèle de corruption adaptative. Ensuite, nous utilisons deux algorithmes de chiffrement asymétrique issus de la cryptographie basée sur les codes correcteurs d'erreur, un dans la métrique de Hamming et l'autre en métrique Rang. En effet, ils sont efficaces et sont résistants aux algorithmes quantiques qui menacent actuellement une partie de la cryptographie traditionnelle. En plus de proposer nos constructions théoriques, nous avons également implémenter et évaluer nos deux versions du protocole, montrant une praticabilité certaine.

Publications et interventions

Interventions

— 2020 : Présentation de l'article *Round-Optimal Constant-Size Blind Signatures* aux journées Cryptographie et Codages (C2), Erdeven, France (évènement virtuel).

— 2020 : Présentation de l'article *Round-Optimal Constant-Size Blind Signatures* à la conférence internationale SECRYPT 2020, Paris, France (évènement virtuel).

— 2020 : Présentation de l'article *CROOT : Code-based Round-Optimal Oblivious Transfer* à la conférence internationale SECRYPT 2020, Paris, France (évènement virtuel).

— 2019 : Présentation d'un travail de groupe sur l'étude de techniques stéganographiques afin de diffuser des maliciels (*Deep Learning Steganography to Hide Malware in Web Content*) en association avec Thales lors des journées REDOCS organisées par le GDR Sécurité Informatique, Gif-sur-Yvette, France.

— 2019 : Présentation de l'article *Efficient and Round-Optimal Secret Handshake* dans le cadre du projet SVP-IoT, Bordeaux, France.

Publications

[ABFG20] CROOT : Code-based Round-Optimal Oblivious Transfer (Nicolas Aragon, Olivier Blazy, Neals Fournaise, Philippe Gaborit), In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2 : SECRYPT, Lieusaint, Paris, France, July 8-10, 2020 (Pierangela Samarati, Sabrina De Capitani di Vimercati, Mohammad S. Obaidat, Jalel Ben-Othman, eds.)*, ScitePress, 2020.

[BBCF20] Round-Optimal Constant-Size Blind Signatures (Olivier Blazy, Laura Brouilhet, Céline Chevalier, Neals Fournaise), In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2 : SECRYPT, Lieusaint, Paris, France, July 8-10, 2020 (Pierangela Samarati, Sabrina De Capitani di Vimercati, Mohammad S. Obaidat, Jalel Ben-Othman, eds.)*, ScitePress, 2020

En cours de soumission

1. *Efficient and Round-Optimal Secret Handshake* avec Olivier Blazy, Céline Chevalier, Emmanuel Conchon, Mathieu Klingler et Olivier Levillain

Table des figures

Bibliographie

[AAB+17a] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillippe Gaborit, and Gilles Zémor. RQC. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.

[AAB+17b] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. HQC. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.

[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.

[Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Heidelberg, May 2001.

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005.

[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.

[AGM+13] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm : a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2) :111–128, 2013.

[AKB07] Giuseppe Ateniese, Jonathan Kirsch, and Marina Blanton. Secret handshakes with dynamic and fuzzy matching. In *NDSS 2007*. The Internet Society, February / March 2007.

[BBB+19] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. An algebraic attack on rank metric code-based cryptosystems. *arXiv preprint cs/1910.00810*, 2019.

[BBC+13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, Heidelberg, February / March 2013.

[BBC+13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, August 2013.

[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.

[BBDQ18] Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 644–674. Springer, Heidelberg, March 2018.

[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.

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of UC-secure oblivious transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15*, volume 9092 of *LNCS*, pages 65–86. Springer, Heidelberg, June 2015.

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 339–369. Springer, Heidelberg, December 2016.

[BCC+09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2009.

[BCG17] Olivier Blazy, Céline Chevalier, and Paul Germouty. Almost optimal oblivious transfer from QA-NIZK. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17*, volume 10355 of *LNCS*, pages 579–598. Springer, Heidelberg, July 2017.

[BCV19] Olivier Blazy, Céline Chevalier, and Quoc Huy Vu. Post-quantum uc-secure oblivious transfer in the standard model with adaptive corruptions. In *Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019, Canterbury, UK, August 26-29, 2019*, pages 28 :1–28 :6. ACM, 2019.

[BDD+17] Paulo S. L. M. Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. A framework for efficient adaptively secure composable oblivious transfer in the ROM. Cryptology ePrint Archive, Report 2017/993, 2017. `http://eprint.iacr.org/2017/993`.

[BDS+03a] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *2003 IEEE Symposium on Security and Privacy*, pages 180–196. IEEE Computer Society Press, May 2003.

[BDS+03b] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196. IEEE Computer Society, 2003.

[BFP+01] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In Ajay D. Kshemkalyani and Nir Shavit, editors, *20th ACM PODC*, pages 274–283. ACM, August 2001.

[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.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 82–99. Springer, Heidelberg, December 2013.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4) :297–319, September 2004.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.

[BPRS17] Megha Byali, Arpita Patra, Divya Ravi, and Pratik Sarkar. Fast and universally-composable oblivious transfer and commitment scheme with adaptive security. Cryptology ePrint Archive, Report 2017/1165, 2017. `https://eprint.iacr.org/2017/1165`.

[BPV12a] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Compact round-optimal partially-blind signatures. In Ivan Visconti and Roberto De Prisco,

editors, *SCN 12*, volume 7485 of *LNCS*, pages 95–112. Springer, Heidelberg, September 2012.

[BPV12b] 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.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical : A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994.

[Bra94] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, Heidelberg, August 1994.

[Can01] Ran Canetti. Universally composable security : A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, Heidelberg, August 1990.

[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.

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Heidelberg, May 2005.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Heidelberg, May 2005.

[CJT04] Claude Castelluccia, Stanislaw Jarecki, and Gene Tsudik. Secret handshakes from CA-oblivious encryption. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 293–307. Springer, Heidelberg, December 2004.

[CKSK08] Alexei Czeskis, Karl Koscher, Joshua R. Smith, and Tadayoshi Kohno. RFIDs and secret handshakes : defending against ghost-and-leech attacks and unauthorized reads with context-aware communications. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 479–490. ACM Press, October 2008.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, Heidelberg, February / March 2013.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CO15] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 40–58. Springer, Heidelberg, August 2015.

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, Heidelberg, August 2003.

[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.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976.

[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.

[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.

[FHS19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2) :498–546, April 2019.

[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.

[FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology — AUSCRYPT '92*, pages 244–251, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[FPV09] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 226–247. Springer, Heidelberg, December 2009.

[Fuc11] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, Heidelberg, May 2011.

[Gha17] Essam Ghadafi. Efficient round-optimal blind signatures in the standard model. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 455–473. Springer, Heidelberg, April 2017.

[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.

[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.

[GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 505–523. Springer, Heidelberg, August 2009.

[GX11] J. Gu and Z. Xue. An improved efficient secret handshakes scheme with unlinkability. *IEEE Communications Letters*, 15(2) :259–261, February 2011.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 111–129. Springer, Heidelberg, August 2007.

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1) :158–193, January 2012.

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019.

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In

Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 491–511. Springer, Heidelberg, December 2014.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

[JKT06] Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Authentication for paranoids : Multi-party secret handshakes. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS 06*, volume 3989 of *LNCS*, pages 325–339. Springer, Heidelberg, June 2006.

[JL07] Stanislaw Jarecki and Xiaomin Liu. Unlinkable secret handshakes and key-private group key management schemes. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 270–287. Springer, Heidelberg, June 2007.

[JL09] Stanislaw Jarecki and Xiaomin Liu. Private mutual authentication and conditional oblivious transfer. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 90–107. Springer, Heidelberg, August 2009.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* Chapman & Hall/CRC, 2nd edition, 2014.

[KMO08] Kazukuni Kobara, Kirill Morozov, and Raphael Overbeck. *Coding-Based Oblivious Transfer*, pages 142–156. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[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.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, Heidelberg, December 2009.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, Heidelberg, March 2011.

[LM18] Baiyu Li and Daniele Micciancio. Equational security proofs of oblivious transfer protocols. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 527–553. Springer, Heidelberg, March 2018.

[MNL16] Yan Michalevsky, Suman Nath, and Jie Liu. Mashable : Mobile applications of secret handshakes over bluetooth le. In *Proceedings of the 22Nd Annual*

International Conference on Mobile Computing and Networking, MobiCom '16, pages 387–400, New York, NY, USA, 2016. ACM.

[MR16] U. Maurer and J. Ribeiro. New perspectives on weak oblivious transfer. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 790–794, July 2016.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

[NP05] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1) :1–35, January 2005.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, August 1993.

[Oka06] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer, Heidelberg, March 2006.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, Heidelberg, August 1992.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3) :361–396, June 2000.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981.

[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.

[SU12] Dominique Schröder and Dominique Unruh. Security of blind signatures revisited. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 662–679. Springer, Heidelberg, May 2012.

[TX06] Gene Tsudik and Shouhuai Xu. A flexible framework for secret handshakes. In George Danezis and Philippe Golle, editors, *PET 2006*, volume 4258 of *LNCS*, pages 295–315. Springer, Heidelberg, June 2006.

[XY04] Shouhuai Xu and Moti Yung. k-Anonymous secret handshakes with reusable credentials. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 158–167. ACM Press, October 2004.

[Yao82a] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[Yao82b] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982.

Résumé

En ayant le respect de la vie privée en tête, nous nous sommes attachés, durant cette thèse, à la conception et à l'amélioration de primitives cryptographiques connues telles que la signature ou l'authentification.
En effet, nous avons proposé un protocole d'authentification (*Secret Handshake*) offrant un anonymat lors de la communication initiale et si celui-ci échoue ; il a la particularité d'être optimal en nombre de tours et efficace en terme de coûts calculatoires et d'échanges.
Aussi, nous présentons une signature dite aveugle, dont le vote électronique est une application quasi-directe. Elle possède diverses propriétés rarement réunies : optimale en nombre de tours, de taille constante en fonction de la taille des messages et prouvée sûre sous des hypothèses classiques dans le modèle standard.
Ensuite, un protocole d'*Oblivious Transfer* en cryptographie à base de codes correcteurs d'erreurs est donné : il offre ainsi une solution résistante à l'ordinateur quantique tout en étant performante en pratique.
Toutes ces constructions sont réalisées avec une approche combinant d'autres outils cryptographiques de manière modulaire mais également en prouvant la sécurité sous des hypothèses de difficulté bien connues.

Abstract

With privacy in mind, we explored, in this thesis, the design and improvement of known cryptographic primitives such as signature or authentication.
Indeed, we propose an authentication protocol, a Secret Handshake, offering anonymity to the users in case of failure ; it is round-optimal and efficient both in computation and communication.
We also present a Blind Signature, e-voting being an almost direct application. It has three properties rarely found combined : round-optimal, constante size and proved secure under classical assumptions in the standard model.
Then, an Oblivious Transfer protocol from Code-based cryptography is detailed giving a quantum-resistant and practical algorithm.
All the constructions were achieved following a modular approach, combining cryptographic building blocks such as encryption, signature and proof systems, and proven secure under known hardness assumptions.