Privacy preserving post-quantum cryptography

Guilaume Kaim

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Guillaume Kaim

Cryptographie post-quantique pour la protection de la vie privée

Thèse présentée et soutenue à Rennes, le 16 décembre 2020
Unité de recherche : IRISA, UMR 6074

Rapporteurs avant soutenance :

Olivier Blazy	Maître de conférence, Université de Limoges
Fabien Laguillaumie	Professeur, Université de Lyon 1

Composition du Jury :

Rapporteurs :	Olivier Blazy	Maître de conférence, Université de Limoges
	Fabien Laguillaumie	Professeur, Université de Montpellier
Examinateurs :	Véronique Cortier	Directrice de Recherche CNRS, LORIA
	Sylvain Duquesne	Professeur, Université de Rennes 1
	David Pointcheval	Professeur, Laboratoire d'Informatique de l'ENS
	Jacques Traoré	Ingénieur, Orange Labs
Dir. de thèse :	Sébastien Canard	Ingénieur, Orange Labs
	Adeline Roux-Langlois	Chargée de Recherche CNRS, IRISA

Table of Contents

Chapter 1

Introduction

Ces dernières décennies ont vu l'émergence de nombreux moyens de communications disponibles à un public de plus en plus large. Cet avènement de la libre circulation des échanges numériques entraîne la génération et le stockage d'un flux très important de données qui nécessite un contrôle d'accès selon différent degrés. C'est dans ce contexte que la cryptographie trouve toute son utilité, en effet celle-ci étant par définition la science de la sécurité de l'information, elle offre de nombreuses possibilités quant à la gestion de ces données. Nous pouvons notamment citer la dissimulation de celles-ci pour toute personne non autorisée, allant jusqu'à un accès différencié selon les droits de chaque utilisateur en passant par la possibilité de s'authentifier auprès de n'importe quelle personne ou service.

Au sein de la cryptographie, il existe plusieurs familles cryptographiques, les deux principales étant la cryptographie symétrique, dite à clé secrète et la cryptographie asymétrique, dite à clé publique. Prenons l'exemple d'un schéma de chiffrement, dans lequel un premier utilisateur veut envoyer un message à un second utilisateur mais sans qu'une tierce personne interceptant ce message ne puisse le lire. Concrètement, le premier utilisateur va utiliser un algorithme de chiffrement pour rendre son message inintelligible, tandis que le destinataire de ce message chiffré va utiliser un algorithme de déchiffrement pour récupérer le message originel. Dans le cadre de la cryptographie symétrique, les 2 parties utilisent une clé commune, qui sera utilisée dans l'algorithme de chiffrement et de déchiffrement. Á contrario, dans un scénario de cryptographie asymétrique, la clé utilisée pour chiffrer le message est distincte de celle utilisée pour l'algorithme de déchiffrement, ce qui permet de rendre la clé de chiffrement publique, puisqu'elle n'a d'autre utilité que de transformer un message en chiffré inintelligible pour toute personne ne possédant pas la clé secrète associée.

Avec la cryptographie asymétrique apparaît la notion très intéressante de sécurité prouvée. En effet, historiquement pour prouver qu'un schéma est sûr, il est soumis à tous les outils disponibles, du point de vue d'un attaquant. Cette variante de la cryptographie, appelée cryptanalyse, a pour but d'essayer de casser les schémas cryptographiques. Cepen-

dant ce type de sécurité n'est pas très formel, puisque les outils composant la cryptanalyse ne sont forcément, pas exhaustifs. C'est pourquoi des preuves de sécurité, reposant sur une technique appelée réduction, ont été mises en place. Concrètement supposons qu'il existe un attaquant qui arrive à casser la sécurité du schéma considéré, et montrons qu'à partir de cet attaquant il est possible d'en construire un second sur un autre problème dont la difficulté est prouvée. Ces preuves de sécurité sont quasiment devenues la norme pour toutes les constructions s'appuyant sur la cryptographie asymétrique, mais peuvent reposer sur des hypothèses très diverses.

En pratique les schémas symétriques obtiennent généralement une meilleure efficacité dans l'exécution des différents algorithmes qui les composent, tandis que les schémas asymétriques autorisent plus de souplesse dans les diverses applications qui les emploient. C'est d'ailleurs ces constructions asymétriques qui vont nous intéresser plus particulièrement tout au long de cette thèse.

1.1 Protection de la vie privée

Comme évoqué précédemment, la sécurité de l'information est devenue un enjeu important dans notre société hyperconnectée. En effet, l'information circule plus activement que jamais et la protection des données de chaque utilisateur est au cœur de nombreuses préoccupations. Cela a entraîné nombre d'organisations et gouvernements à prendre en considération ces enjeux, par exemple au travers du règlement général sur la protection des données (RGPD ou GDPR en anglais), qui a défini de nombreuses règles à respecter pour garantir cette protection de la vie privée au sein des pays de l'Union Européenne.

Il existe aussi des primitives cryptographiques garantissant ce respect de la vie privée des utilisateurs, pouvant se déployer dans des cas d'usages où la manipulation de ces données nécessite une vigilance accrue sur l'utilisation qui en est faite. Ces primitives ont été continuellement introduites et améliorées au cours des 50 dernières années, permettant notamment à de nombreux utilisateurs d'échanger et de transmettre des données pouvant être à caractère personnel, et donc sensibles, sans pour autant craindre que celle-ci soient accessibles à n'importe qui.

On retrouve parmi ces constructions, des schémas de chiffrement comme évoqués précédemment, tel que le chiffrement homomorphe, qui permet d'effectuer n'importe quel type de calcul sur des données chiffrées, mais également des schémas de signatures, permettant à un utilisateur de s'authentifier par la génération d'une signature numérique sur

un message donné d'une manière anonyme. De la même façon que pour le chiffrement, il existe des schémas de signature plus complexes, dans le cadre de la protection de la vie privée, par exemple les signatures de groupes, qui permettent à un utilisateur de générer une signature au nom d'un groupe, sans pour autant divulguer son identité exacte au sein de ce groupe.

À ce jour, il existe des constructions de telles primitives possédant une efficacité tout à fait satisfaisante, mais ces constructions risquent d'être menacées dans le futur. En effet, ces schémas reposent sur des problèmes mathématiques, qui sont très difficiles à résoudre pour les ordinateurs actuels, mais qui ne ne représenteront pas une telle barrière pour des ordinateurs quantiques, comme démontré par Shor [Sho97]. En conséquence, des solutions doivent être envisagées pour pouvoir répondre à cette problématique.

1.2 Cryptographie classique et post-quantique

Ces nouvelles solutions doivent s'appuyer sur des problèmes mathématiques qui ne sont pas vulnérables à l'avantage qu'ont les ordinateurs quantiques sur les primitives cryptographiques se basant sur des problèmes mathématiques dits "standards". Ces outils mathématiques "standards" se composent notamment du cryptosystème RSA [RSA78] (du nom de ses auteurs Rivest, Shamir et Adleman) qui repose sur le problème de la factorisation, ou du cryptosystème El-Gamal reposant sur le problème du logarithme discret.

Dans le cas des cryptosystèmes de type RSA, la clé secrète est composée entre autre de deux nombres premiers p et q dont leur produit $N = p \times q$ fait partie de la clé publique. Un attaquant qui arriverait donc à factoriser le nombre N pour retrouver les deux nombres premiers p et q, pourrait attaquer la sécurité des schémas de type RSA.

Concernant les cryptosystèmes basés sur le logarithme discret, on considère un groupe $\mathcal{G}$ bien choisi, le plus souvent d'ordre premier p. On va ensuite choisir aléatoirement un entier a comme partie de la clé secrète et son pendant publique qui lui est associé est g^a avec g un élément choisi aléatoirement dans le groupe $\mathcal{G}$. On définit alors le problème du logarithme discret, demandant de retrouver l'entier a à partie de l'élément g^a appartenant au groupe $\mathcal{G}$.

Ces deux types de problèmes ont été à l'origine de nombreux schémas cryptographiques sûrs et proposant une efficacité très attractive mais, comme énoncé précédemment, ils seront vulnérables dans une ère post-quantique. En effet, conserver leur sécurité impli-

querait d'augmenter considérablement leurs paramètres afin de pouvoir résister à une attaque basée sur des ordinateurs quantiques. On leur préfèrera dès lors des cryptosystèmes, dits "résistants au post-quantique", pour lesquels nous n'avons à ce jour pas découvert d'avantage pour un ordinateur quantique par rapport à un ordinateur classique.

Dans ce contexte, une compétition lancé par l'institut de standardisation américain (NIST) a été lancé en 2016 afin de proposer de telles alternatives dans un monde où les ordinateurs quantiques seraient une réalité quotidienne. Cette compétition a créé une effervescence dans le milieu de la recherche cryptographique, afin de proposer un cryptosystème proposant toutes les caractéristiques nécessaires pour devenir un standard de la cryptographie post-quantique. Comme candidats proposés, nous pouvons citer les cryptosystèmes basés sur les codes correcteurs, sur les isogénies ou encore sur les polynômes multivariés. Cependant ce sont les constructions basées sur les réseaux euclidiens (dits *lattices* en anglais) qui vont nous intéresser plus particulièrement ici.

1.3 Réseaux euclidiens

Les réseaux euclidiens peuvent se décrire de façon simple, comme un ensemble infini de points répartis de manière ordonnée dans l'espace. D'un point de vue plus mathématiques, on peut le définir comme étant un sous-groupe discret engendré par une base de vecteurs de l'espace. L'un des avantages considérable des réseaux euclidiens en cryptographie, vient du fait que les opérations effectuées se composent essentiellement de produits matrices-vecteurs qui sont faciles à comprendre et à implémenter en pratique. Cependant afin de garantir la sécurité des constructions, les matrices et vecteurs considérés doivent être composés de plusieurs milliers de lignes et/ou colonnes, ce qui affaiblit souvent leur efficacité.

Concernant l'aspect sécurité, on peut noter que les problèmes basés sur les réseaux euclidiens se décomposent en 2 niveaux, qui n'auront certes pas la même finalité mais qui seront tout autant nécessaires pour prouver la sécurité des cryptosystèmes. Le 1er niveau se compose principalement du problème SVP (Shortest Vector Problem) ou problème du vecteur le plus court, et du problème CVP (Closest Vector Problem) ou problème du vecteur le plus proche. Concernant le problème SVP, considérons une base aléatoire d'un réseau euclidien, il demande alors d'en donner le vecteur le plus court, tandis que dans le problème CVP, étant donné un point quelconque de l'espace, il demande de trouver le point du réseau euclidien qui en est le plus proche. Ces deux problème ont été dérivés en

diverses variantes, notamment en autorisant divers facteurs d'approximation.

Cependant ces problèmes ne permettent pas de produire directement des constructions cryptographiques, et sont plutôt utilisés pour prouver la difficulté des problèmes qui composent le 2ème niveau qui eux, seront utilisés comme base de cryptosystèmes. Dans cette famille, on distingue tout d'abord le problème LWE (*Learning With Errors*), qui étant donné un secret $\mathbf{s} \in \mathbb{Z}_q^n$ et une matrice $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ sur-déterminée, va ajouter une petite erreur $\mathbf{e}$ au produit $\mathbf{A} \cdot \mathbf{s}$ afin de générer un échantillon LWE. Le problème se dérive en 2 variantes, la première demande de retrouver le secret $\mathbf{s}$ à partir d'un échantillon LWE $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \mod q$ et de la matrice $\mathbf{A}$, tandis que la deuxième variante demande de décider, étant donné $\mathbf{A}$, si un élément $\mathbf{b}$ a été généré aléatoirement, ou sous la forme d'un échantillon LWE. De plus on peut noter qu'un aspect particulièrement intéressant en cryptographie provient du fait que ces deux variantes sont de difficulté équivalente. Le problème LWE est principalement utilisé pour construire des schémas de chiffrement, allant d'un chiffrement à clé publique tout à fait classique [Reg05] à un chiffrement homomorphe bien plus complexe [Gen09].

Le deuxième problème qui est utilisé pour construire des primitives cryptographiques est le problème SIS, qui étant donné une matrice $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ sous-déterminée demande de produire un petit vecteur $\mathbf{v} \in \mathbb{Z}_q^m$ vérifiant l'équation $\mathbf{A} \cdot \mathbf{v} = 0 \mod q$. Ce type de problème est majoritairement utilisé pour construire des schémas de signature, nous pouvons d'ailleurs différencier 2 types de modèles de construction, le premier introduit un système de trappe sur une fonction associée au problème SIS, qui est fournie au signeur, lui permettant ainsi d'effectuer des opérations qui lui seraient impossibles en son absence. Tandis que le deuxième modèle appelé "signature Fiat-Shamir"transforme un schéma interactif appelé schéma d'identification en remplaçant une partie de l'interaction par une fonction modélisée de telle sorte que sa sortie soit considérée aléatoire. Dans ces deux modèles, la signature est un vecteur solution d'un problème SIS, ce dernier étant public. C'est notamment à partir de ces constructions que seront articulées la majorité des contributions qui sont présentées dans cette thèse.

1.4 Contributions

Dans la suite de cette introduction, je vais détailler les contributions que j'ai pu apportées durant mes 3 années de thèses, qui concernent principalement la conception de schémas cryptographiques respectueux de la vie privée, ainsi que l'adaptation en cas

d'usage de l'un d'entre eux. Toutes ces constructions s'appuient sur la cryptographie basée sur les réseaux euclidiens décrite ci-dessus.

1.4.1 Signatures de groupes

Ma première contribution concerne la construction d'une signature de groupe sur les réseaux euclidiens, à laquelle nous ajoutons quelques propriétés attractives. Par définition, une signature de groupe est un schéma permettant à un groupe d'utilisateurs de signer au nom de ce même groupe, tout en restant anonyme parmi ses membres. Elles fournissent une caractéristique supplémentaire qui est de permettre "l'ouverture" d'une signature pour en retrouver le membre du groupe qui en est à l'origine. Ces schémas ont de nombreuses applications dans la vie courante. Prenons par exemple un cas où une personne doit prouver qu'elle a un certain âge (par exemple moins de 25 ans pour pouvoir prétendre à une réduction), il suffirait qu'elle produise sa carte d'identité qui contient sa date de naissance et donc son âge. Cependant en produisant ainsi sa carte d'identité, elle fournit par la même occasion son nom, son adresse..., alors que seul le fait qu'elle appartient au groupe des moins de 25 ans est nécessaire. Les signatures de groupes permettent donc d'éviter de tels désagréments.

Il existe à ce jour de nombreux schémas de signature de groupe, mais les plus efficaces d'entre eux s'appuient sur des hypothèses qui ne résisteraient pas à un ordinateur quantique, tels que le logarithme discret défini précédemment. C'est pourquoi depuis une dizaine d'années, un grand nombre de constructions basées sur les réseaux euclidiens ont été conçues ([Gol+10; Lag+13; Lan+14; Lib+16b; Lin+17]...). Ces constructions doivent cependant se plier à l'inefficacité, sur les réseaux, des preuves de connaissance à divulgation nulle de connaissance nécessaires pour de telles constructions dans le modèle habituel de Bellare et al [BMW03]. Cependant en 2019, un nouveau schéma publié à Eurocrypt par Kastumata et Yamada [KY19] s'écarte de ce modèle habituel. Leur construction s'appuie sur des primitives plus récentes telles que les signatures basées sur les attributs pour court-circuiter les problèmes d'inefficacité, en plus de reposer sur un modèle de sécurité plus commun.

À partir de cette nouvelle construction nous proposons un schéma de signature de groupe basé sur les réseaux qui bénéficie des mêmes avantages que le schéma de [KY19] et nous ajoutons en plus une nouvelle propriété appelée *forward security*. Celle-ci permet de diviser le temps en plusieurs périodes, de sorte que les clés secrètes de tous les membres du groupe sont mises à jour à chaque nouvelle période de temps sans possibilité de récupérer

une clé secrète pour une période de temps passée. L'intérêt majeur de cette propriété est d'éviter que les signatures déjà générées ne soient rendues invalides dans le cas où la clé secrète d'un membre du groupe est volée. En effet dans un schéma de signature de groupe classique, la clé secrète ne change pas au cours du temps, et dans le cas de figure où un utilisateur se fait subtiliser sa clé secrète il devient alors impossible de s'assurer qu'une signature de groupe a bien été générée par un utilisateur certifié. La propriété de *forward security* permet donc de palier à ce problème. Nous développons ce travail dans le chapitre 4 de cette thèse.

— [CGKRT20] Sébastien Canard, Adela Georgescu, Guillaume Kaim, Adeline Roux-Langlois and Jacques Traoré. Constant-size lattice-based group signature with forward secrecy in the standard model. Published in *PROVSEC 2020*.

1.4.2 Signature aveugle

Les signatures aveugles ont été introduites en 1982 par Chaum [Cha82] pour le cas d'usage précis de la monnaie électronique. L'idée d'une signature aveugle est la suivante: un utilisateur veut générer une signature sur un message de son choix, cependant il ne possède pas la clé secrète lui permettant de générer une telle signature. Pour palier ce problème, il a la possibilité d'interagir avec une autorité qui possède cette clé secrète et qui est alors vu comme un signeur. Cependant l'utilisateur requiert que la signature qu'il va sortir à la fin de l'interaction avec le signeur ne puisse pas être tracée par ce dernier. Pour en revenir au cas d'usage de la monnaie électronique, ce type de schéma permet donc à un utilisateur de faire certifier de la monnaie électronique à une autorité (une banque par exemple), sans que celle-ci puisse retrouver l'identité de l'utilisateur lorsque celui-ci la dépense.

Concernant les constructions existantes, elles sont nombreuses et très efficaces sur la cryptographie "classique". Nous pouvons notamment citer le schéma Blind RSA [Fer93] qui se base sur le cryptosystème RSA, tandis que l'histoire des signatures aveugles sur les réseaux euclidiens est bien plus complexes. Tout d'abord il n'existe qu'un seul modèle de construction basé sur un schéma d'identification de Lyubashevsky [Lyu08]. Ce modèle requiert 3 échanges entre l'utilisateur et le signeur, comme d'autres schémas tel que Okamoto-Schnorr [Oka92], à contrario d'autres constructions comme blind RSA qui n'en demandent que 2.

De plus toutes les constructions sur les réseaux euclidiens ont vu leur preuve de sécurité

invalidées par un papier de Hauck et al [Hau+20] publié à Crypto 2020. En effet pour prouver qu'un utilisateur malhonnête ne peut pas générer plus de signatures qu'autorisées, une propriété appelée *one-more unforgeability* est nécessaire. Une modèle de preuve a été proposé par Pointcheval et Stern [PS00] pour des constructions basées sur des groupes finis, dans lesquels un élément secret était choisi aléatoirement dans tout le groupe. La différence sur les réseaux euclidiens vient du fait qu'une clé secrète est habituellement un petit élément de l'ensemble considéré, ce qui interdit toute adaptation directe sur les réseaux euclidiens de la preuve de Pointcheval et Stern.

Dans notre construction, nous revenons sur cette preuve et sur la solution apportée par Hauck et al. [Hau+20] qui souffre d'un manque d'efficacité. Concernant notre construction, notre sécurité repose sur une conjecture qui n'a pas été prouvée. Tout d'abord, notre solution gagne en efficacité en utilisant l'avantage de distributions gaussiennes sur des distributions uniformes. De plus, nous utilisons des outils tels que les trappes sur les réseaux euclidiens pour effacer toute trace d'abandon durant le protocole de signature aveugle, notre construction étant la seule n'ayant pas recours aux abandons, qui est un impact négatif sur l'efficacité. Finalement, nous proposons une variante appelée signature partiellement aveugle, qui permet à l'utilisateur et au signeur d'inclure une information publique, sur laquelle ils se sont mis d'accord en amont de la génération de la signature. Notre construction n'entraîne quasiment aucun coût supplémentaire par rapport à la signature aveugle. Ce travail est décrit dans le chapitre 5 de cette thèse.

— [BCEKRT20] Samuel Bouaziz-Ermann, Sébastien Canard, Gautier Eberhart, Guillaume Kaim, Adeline Roux-Langlois and Jacques Traoré. Lattice-based partially blind signature without restart. Cryptology ePrint Archive, Report 2020/260, 2020. https://eprint.iacr.org/2020/260.

1.4.3 Vote électronique

Finalement, nous avons aussi construit un schéma de vote à partir de notre signature aveugle, en suivant le procédé de Fujioka et al. [FOO92]. Habituellement les schémas de vote électronique utilisent à leur avantage les propriétés homomorphiques de certaines primitives cryptographiques pour améliorer l'efficacité de certains aspects de leur schéma de vote. Cependant ces primitives bénéficiant de propriétés homomorphiques sont aussi sujettes à des soucis d'inefficacité (tel que l'utilisation de preuves de connaissance, coûteuses sur les réseaux euclidiens). C'est pourquoi il est important de diversifier les modèles de

constructions de tels schémas pour pouvoir adapter chaque construction à chaque besoin.

Sur les réseaux euclidiens, il existe actuellement 2 constructions de vote électronique par Chilloti et al. [Chi+16] et del Pino et al. [Pin+17]. Ces constructions reposent sur des outils utilisant des propriété homomorphiques (chiffrement complètement homomorphe pour la première et commitment homomorphe avec preuve de connaissance pour la seconde), tandis que notre construction repose sur d'autres primitives.

Notre contribution est une adaptation du schéma de vote [FOO92] au post quantique.

1. Elle utilise comme base le schéma de signature aveugle que nous avons décrit juste au-dessus.
2. Nous lui ajoutons un schéma de chiffrement qui viendra remplacer le schéma de commitment utilisé initialement dans le modèle de [FOO92], afin d'éviter que tout les votants aient la nécessité d'être en ligne à la fin de l'élection pour aider à ouvrir leurs bulletins.

De plus nous adaptons nos primitives pour leur rajouter la propriété *threshold* (dit à seuil), qui permet de distribuer les clés secrètes entre diverses autorités pour éviter le cas problématique où une autorité corrompue possède tous les pouvoirs pour pouvoir altérer l'élection. Nous prouvons par ailleurs la sécurité de notre schéma de vote à partir de la sécurité des primitives sous-jacentes qui la composent.

Les avantages de notre construction par rapport aux schémas existants reposent sur le fait que certaines opérations (telle que la vérification qu'un bulletin de vote est bien valide) sont plus rapides dans notre construction. De plus l'efficacité de notre construction ne s'affaiblit pas avec le nombre de candidats considérées, contrairement aux schémas de vote basés sur des primitives bénéficiant de propriétés homomorphiques. Cependant d'autres opérations (telle que la phase de comptage) sont plus rapides dans les constructions mentionnées précédemment.

Ce travail disponibles dans le chapitre 6, n'a pas encore été rédigé sous forme d'article et ne fait donc pas encore l'objet d'une soumission.

Chapter 2

Introduction in english

Every year, the amount of communication all around of the world is growing and accessible to a larger public. However this emergence of digital exchanges leads to the birth and storage of many data that can potentially be sensitive and then must be protected from the potential abuse that can be done on them. In order to prevent such attacks, cryptography is a good answer since by definition it is the science of the security of information, then it produces tools that have been designed to protect such data at different levels. These levels starts, from a simple encryption scheme, permiting to anyone to hide a given message to everyone but those who owns the corresponding secret decryption keys. A stronger level is the homomorphic schemes allowing to additionally perform some operations on encrypted data, without decrypting them. We can also cite the signature schemes, where the possessor of a secret key associated to a public key can authenticate himself as the owner of this corresponding secret key.

Cryptography can be split in several families, with the two mains ones being the symmetric and asymmetric cryptography. The first one is also called secret key cryptography, since if we consider an encryption scheme for example, the key used to encrypt a message into a ciphertext and the key used to decrypt this same ciphertext to recover the message are the same and must remain private. Concerning the asymmetric cryptography, that is also called public key cryptography, considering the same example of encryption scheme, the key used to encrypt and the key used to decrypt are not the same, the former is called the public key, while the later is called the secret key. As their name mentioned it, the public key can be revealed publicly to anyone, while the secret key must be kept secret to everyone but those granted with the right to decrypt. The differences on these two cryptography are numerous but in resume, the symmetric cryptography gets a better efficiency, while the asymmetric cryptography offers more possibilities and applications in general.

An important feature of the asymmetric cryptography, is the notion of provable security that is now close to become a norm for every new designed scheme. Historically, the

robustness of a cryptographic scheme was measured by its ability to resist to every known attack. Indeed, the corresponding science, called cryptanalysis, was the only way to ensure that a construction is safe or not. However, since the tools composing such cryptanalysis are not exhaustive, the possibility that a new attack comes up within time still remain possible. In public key cryptography, the way to prove the security of a construction is different. This new way makes use of a specific type of proof called reduction. It means that to prove that a given cryptographic scheme is secure, we assume the opposite, and prove that it would imply a second attacker on another problem that has already been assumed to be proven secure. We can then conclude that the former scheme is indeed secure.

2.1 Cryptography for privacy

As already discussed, the security of the information has become a major challenge for all governments and many industries. Indeed, the amount of communications leads to a privacy concern about the data moving during these exchanges, this concern is shared by every party included in these interactions. Concretely, some measures have already been carried out, especially the General Data Protection Regulation set up by the European countries to protect the privacy of every European citizen.

In this context of privacy, cryptography brings some very useful tools that can guarantee different variants of privacy depending on the needs. Indeed, several cryptographic primitives especially designed to work in a privacy protection environment have been introduced and improved over the last decades. These primitives allow the users to digitally send sensitive data without be worried that anyone can get an access to them, this access is limited to the authorized persons or services.

Examples of such primitives are numerous, the most famous being homomorphic encryption. This primitive allows a user to encrypt some data, such that authorized entities can perform some computation on them, while the data remains encrypted during all the process, ensuring then the confidentiality of this same data. In this thesis we take a closer look on digital signatures schemes, that allow the owner of a dedicated secret key to be authenticated. The concept of digital signature can be derived in group signature for example, where a group of user gather, such that any member of this group can anonymously sign on behalf of the group.

There exists efficient constructions that are built upon these theoretical primitives,

relying on mathematic problems that are very hard to solve for the current computers. However all these cryptosystems are harmed in the future with the rise of the quantum computer, that profit of significant advantage compared to current one, These advantages were revealed by Shor [Sho97].

2.2 Classical and post-quantum Cryptography

In response to this threat, new cryptosystems must be built upon mathematical problems not weakened by quantum computer. Indeed, the most famous cryptosystem, known as RSA, is called from the name of its inventors (Rivest, Shamir, Adleman) and is based on the factorization problem, we can also cite the El-Gamal cryptosystems based on the discrete logarithm problem, that would also be harmed by quantum computers.

Concerning the RSA cryptosystem, the secret key includes two big primes p and q, while the public key contains their product $N = p \cdot q$. Then if an attacker can factorize N to recover p and q, it means that it would break the RSA cryptosystem.

For the El-Gamal cryptosystem, using a well chosen group $\mathcal{G}$, the secret key is an integer $a \in \mathbb{Z}$, such that for a random element $g \in \mathcal{G}$ the public key component is g^a. Finally the discrete logarithm problem asks to find a from g^a.

A lot of cryptographic schemes, that are very efficient and used in practice, are using these two problems. However they can not resist to a quantum computer and would be broken in a quantum era as shown by Shor [Sho97]. The question now is then, which cryptosystems can replace them to produce the new standards when the quantum computers will become a reality. Indeed, if we would like to keep these cryptosystems in practice, it would imply to enlarge so much the parameters to remain secure. This is why it would be better to find new mathematical problems for which the quantum computer will have no more advantages compared to current ones.

To answer this question, the NIST has proposed a competition to find new standards in a world, where the quantum computers are a reality. Taking a closer look into this competition, we can shed light upon the lattices-based constructions that are the most represented cryptosystem among others like the code-based or the isogeny-based cryptography. Indeed, the lattices interest us specifically, as they compose the basis of the work described in this thesis.

2.3 Lattices

Lattices can simply be described as an infinite set of points well distributed in the space, while in a more mathematical point of view the lattices are discrete sub groups generated by a basis of vector. Lattices have numerous advantages, compared to others post-quantum cryptosystems proposed, like the ease of use since the computations are only composed of matrix vectors products or additions. It means that the understanding of the operations composing all constructions are simple to understand, while the implementation of these same operations are easy to perform. The shortcoming being that the size of these vectors and matrices are huge since composed of thousands elements, making lattice-based cryptography suffering a lack of efficiency at first sight.

However if we consider the security aspect, the topic is more encouraging since there exists very well understood reduction upon lattice problems. These lattice problems can be categorized into two families. The first one is composed of problems that are directly linked to the lattices and is composed of the SVP (for shortest vector problem) and CVP (for closest vector problem) problems. The SVP problem asks, given a lattice, to find the shortest vector composing this lattice, while the CVP problem asks, given any point in the space, to find the point in the lattice that is the closest to this given point. Moreover these two problems can be derived in several variants, asking for example to solve them with an approximation factor.

However these problems cannot directly be used to build cryptographic primitives. They are useful to prove the difficulty of a second family of problems. A first problem is called LWE (Learning With Errors), that asks, given a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and an element $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}, \mathbf{b} \in \mathbb{Z}_q^m$ with $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e}$ being a small error vector, to find the element $\mathbf{s}$. The aforementioned problem is called search LWE but it has another version called decisional LWE which asks, given $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and an element $\mathbf{b} \in \mathbb{Z}_q^n$ to decide if $\mathbf{b}$ is generated uniformly random or as an LWE sample. Whatever the variant, the LWE problem is mainly used to build encryption schemes, and it is on this problem that the first ever fully homomorphic encryption scheme has been built.

The second problem used to build cryptographic primitives on lattices is the SIS (Shortest Integer Solution) one. In this problem, we get a randomly distributed matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and the problem is to find a short vector $\mathbf{v} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \mathbf{v} = 0$. There is a variant called ISIS, where the element on the right of the equality is a public non zero element $\mathbf{x} \in \mathbb{Z}_q^n$ such that we want to solve $\mathbf{A} \cdot \mathbf{v} = \mathbf{x}$. Contrary to the LWE problem, the

SIS problem is mainly used to build digital signature schemes, for which we can describe two types of such constructions. The first one is called Fiat-Shamir signatures and is built from an identification scheme by replacing the interactive part with a hash function modeled as a random oracle, as introduced by Fiat and Shamir [FS86]. The second type of signature [GPV08] relies on a system of trapdoor. Indeed the signer owns a trapdoor as a small basis of the lattice, allowing him to find short vector solutions of an ISIS equation. Our contributions make use of these two types of signatures in their constructions.

As said above, the efficiency of the lattice-based cryptography is far from the one obtained using "classical" cryptography, mainly because of the size of the matrices and vectors that counts more than several thousands of elements. This is why new variants of the LWE and SIS problems have been designed, replacing the integer ring that was initially considered with polynomial rings that can save a factor n (1024 or 2048 in practice) in the size of matrices and vectors. These variants are called Ring LWE and Ring SIS, while the formulation of the problems remains as stated above the elements are not lying anymore in $\mathbb{Z}_q$ but in polynomial rings.

2.4 Contributions

Now, I am going to describe my contributions corresponding to the work I did during my PhD. These contributions are centered around the concept of cryptography for privacy, as we designed several primitives providing privacy, as well as a direct use-case which is the electronic voting.

2.4.1 Group Signatures

To begin with my contributions, I start by describing a lattice-based group signature that adds some attractive properties. Group signatures is a very appealing primitive, since it allows several users to gather into a group such that every member can sign on behalf of this group with the additional property of traceability. Indeed, in case of abuse, a signature generated by a group member can be opened by a manager to recover the identity of the corresponding group member. Moreover the group signature must verify the correctness property, meaning that a genuinely generated group signature must be valid with at least an overwhelming probability. Finally, the group signature must verify the anonymity property, such that anyone, but the group manager, can not recover the

identity of the member that has generated a given signature.

Currently, there is a lot of group signatures scheme that has been designed in "classical" cryptography. However as mentioned earlier these constructions will not remain secure during a quantum era. This is why the topic of finding an efficient post-quantum group signature attracts researchers. Indeed, they focus on designing such a scheme in the lattice settings from a decade with the former work of Gordon et al. [GKV10]. An intense line of research follows this work ([Lag+13; Lan+14; Lib+16a; San+18]...), with the efficiency being improved in each new construction. However, they all face a limitation due to the inefficiency of zero-knowledge proofs that are the building blocks of the general framework of group signatures [BMW03]. In light of this limitation, Katsumata and Yamada departed from the traditional framework by publishing at Eurocrypt 2020 a new construction based on a different framework [KY19]: their construction is based on a recent primitive called attribute-based signature to cut off the aforementioned limitations.

Starting from this work, we build our own group signature scheme that enjoys the attractive property of forward security. Concretely, this property cuts the time into periods such that, when entering a new period, the group secret keys of every member will be updated. In the same time, the secret keys of past periods will be erased by each group member to forbid the generation of a group signature corresponding to a past period of time. Indeed in a group signature not benefiting the forward security, in case of any group secret key being stole, the whole group signature scheme has to be rebuilt. The reason is that there is no way for a verifier to tell if a group signature has been generated with a stolen secret key or the secret key of a certified group member. Thanks to the forward security property, this problem does not apply.

— [CGKRT20] Sébastien Canard, Adela Georgescu, Guillaume Kaim, Adeline Roux-Langlois and Jacques Traoré. Constant-size lattice-based group signature with forward secrecy in the standard model. Published in *PROVSEC 2020*.

2.4.2 Blind signature

We now describe a blind signature that we designed in the lattice-based settings. Blind signatures have been introduced by Chaum in 1982 [Cha82] to the concrete use case of e-cash. In a nutshell, a blind signature scheme is an interactive protocol between a user and a signer, where a user wants to obtain a signature on a message M of its choice, but does not own the corresponding signing secret key. It means that he has to interact

with an authority which possesses this signing key to generate and output the desired signature. However coming back to the e-cash use-case, where the signing authority being a bank for example, and the user being one of its customer, the blind signature process is used to generate an electronic coin to be spent by the customer, but this latter does not want his payment to be traced, this is why he wants the notion of blindness such that the signing authority is not able to link a blind signature to its corresponding transcript. In the same time, coming back to our example, the bank does not want the user to spent more money that authorized. In order to ensure this property the blind signature must verify the one-more unforgeability property guaranteeing that a user can not generate more signatures than the number of interactions he had with the signing authority.

In practice there exists a lot of blind signature constructions, but most of them are built upon classical cryptography. We can cite the most famous one as the Blind RSA scheme [Fer93] that takes as a basis the RSA cryptosystem. Coming to the post-quantum setting, the existing constructions are less numerous. For example in the lattice setting, all the constructions take as a starting point the identification scheme of Lyubashevsky [Lyu08], to get an interactive protocol with 3 exchanges, like the Okamoto-Schnorr construction on the discrete logarithm problem, contrary to the aforementioned Blind RSA scheme that includes only 2 exchanges.

Moreover, history of lattice-based blind signatures is complicated, since the recent paper of Hauck et al. at Crypto 2020 [Hau+20] found a flaw in every previous lattice-based blind signature scheme. This flaw appears in the one-more unforgeability proof, that follows the seminal work of Pointcheval and Stern [PS00] in every known lattice-based construction. Indeed, the [PS00] proof was originally designed for constructions in which the secret key was picked randomly in a whole set, while in lattice cryptography, the secret key is usually a short element of a given set. This difference incurs a correctness error in the lattice-based constructions that is missing in their counterparts on finite group. Thus a direct transformation into the lattice setting is forbidden and needs an adaptation as developed by Hauck et al. [Hau+20].

Coming into our construction, we propose a new blind signature scheme, while our security is only conjectured and not proved. Our scheme relies on the advantages given by the gaussian distribution over the uniform one. Moreover, we succeed to get rid of the restarts existing in all previous schemes by enabling the signer to possess a trapdoor to prevent a signature generation to restart from scratch after a test fails. Finally we also propose a partially blind variant, that allows the user and signer to agree on a common

information that is made public, our construction comes with no more cost compared to the blind signature scheme since there is no more element that are generated or exchanged during the protocol execution.

— [BCEKRT20] Samuel Bouaziz-Ermann, Sébastien Canard, Gautier Eberhart, Guillaume Kaim, Adeline Roux-Langlois and Jacques Traoré. Lattice-based partially blind signature without restart. Cryptology ePrint Archive, Report 2020/260, 2020. https://eprint.iacr.org/2020/260.

2.4.3 E-voting protocol

We finally developed an electronic voting system that takes as a basis the blind signature described above within the framework of Fujioka et al. [FOO92]. Usually electronic voting schemes make use of cryptographic primitives with homomorphic properties to benefit a higher efficiency on some aspect of the voting protocol. However these homomorphic-like constructions come at some cost on other aspects (as the efficiency of the underlying homomorphic construction) and needs other primitives that remains inefficient in the lattice-based settings (like the zero-knowledge proofs). These limitations show that it is important to diversify the considered framework to be adapted for every needs of electronic voting protocols.

Our construction is built in the lattice setting, and prior to our contribution, 2 earlier electronic voting protocols in the lattices were designed. The first one by Chilloti et al [Chi+16] uses fully homomorphic encryption schemes to build their voting protocol. They avoid to use the zero-knowledge proofs, but rely on a very heavy primitive as the fully homomorphic encryption scheme. The second electronic voting scheme on lattices has been designed by del Pino et al. [Pin+17] and mix the use of homomorphic commitment schemes with approximate zero-knowledge proofs that benefit a better efficiency than their exact counterparts, moreover they implemented their construction such that it is possible to represent how their protocol could behave in a real election.

Taking a closer look on our construction, we already said that we follow the [FOO92] framework, but we apply some modifications. Indeed, the original framework makes use of a commitment scheme to hide the voting choice of every voter until the end of the election, where they all open their commitment. However the fact that each voter needs to open individually its commitment induces a lack of efficiency since they need to be online at the end of the election to open its ballot and allow the tally of the election.

We decide to use instead an encryption scheme such that a single secret key is needed to open all the ballots, while the hiding of every voting choice is ensured until the end of the election. We also modify the cryptographic primitive we make use to transform them into threshold variants. Indeed when coming into such a sensitive topic that is the voting protocol, we cannot allow a single authority to possess all the secret keys and then to be able to affect the election freely. This is why we add a secret sharing mechanism such that several authorities are mandatory to perform the signature generation protocol and to rebuild the decryption key of the electronic voting protocol.

We finally mention that our work is still in progress and then has not yet been subject to any submission.

Chapter 3

Preliminaries

We define in this first chapter some basic definitions and tools we are using in the rest of the thesis. We start by defining generic mathematics objects, like the set of elements we work with as well as the basic mathematical tools that will serve all along the chapters composing this thesis. We also define what is a lattice and what are the associated hard problems, from which the lattice-based cryptography is built. We finally introduce some basic cryptographic primitives that will be used as building blocks in the various contributions presented in this thesis.

3.1 Mathematical definitions

We define here the mathematical tools that we make use in the different works presented in the following chapters.

3.1.1 Rings and notations

We denote $[a, b]$ the set of all integers between a and b, while $[a]$ denotes the set of all integers between 1 and a. We denote the integer rings $\mathbb{Z}_q$ with q a prime number, we define as well its polynomial variant $\mathrm{R} = \mathbb{Z}[x]/(f(x))$ with f a polynomial of degree n that will be used for efficiency reason with $\mathrm{R}_q = \mathrm{R}/q\mathrm{R}$.

The vectors are written in bold lower-case letters $\mathbf{a}$, and matrices in bold upper-case letters $\mathbf{A}$. The euclidean norm of a vector is denoted by $\|\mathbf{b}\|$, and the norm of a matrix $\|\mathbf{T}\| = \max_i \|\mathbf{t}_i\|$, which is the maximum norm of its column vectors. The infinite norm $\|\mathbf{a}\|_\infty = \max_i |a_i|$ is the maximal value of its coefficients.

We denote by D a distribution over some countable support S and $x \hookleftarrow D$ the choice of x following the distribution D. Considering that D_1 and D_2 are two distributions over a same countable support S, then we can define their statistical distance by $\Delta(D_1, D_2) = \frac{1}{2}\sum_{x\in S}|D_1(x) - D_2(x)|$.

A function $f(n)$ is said negligible if it exists N_{poly}, such that for all $n > N_{\text{poly}}$, $f(n) < \frac{1}{\text{poly}(n)}$ for every positive polynomial $\text{poly}(n)$.

3.1.2 Probabilities definitions

Let S be a set of size t, the uniform distribution on this set means that every element s $\in$ S has the same probability $\frac{1}{t}$ to be picked. We denote s $\leftarrow_{\$}$ S to indicate that s is uniformly picked in S.

Gaussian distribution. The Gaussian function of center $\mathbf{c} \in \mathbb{R}^n$ and width parameter σ is defined as $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\frac{\|\mathbf{x}-\mathbf{c}\|^2}{\sigma^2})$, for all $\mathbf{x} \in \mathbb{R}^n$. A positive definite covariance matrix is defined as $\mathbf{\Sigma} = \mathbf{B}\mathbf{B}^T$:$\rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}} = \exp(-\pi(\mathbf{x}-\mathbf{c})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c}))$. The discrete Gaussian distribution over a lattice Λ is defined as $D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\sigma,\mathbf{c}}(\mathbf{\Lambda})}$ where $\rho_{\sigma,\mathbf{c}}(\mathbf{\Lambda}) = \sum_{x\in\Lambda}\rho_{\sigma,\mathbf{c}}(\mathbf{x})$.

3.2 Lattices

3.2.1 Definition of a Lattice

We define a n-dimensional full rank lattice Λ as a discrete additive subgroup of $\mathbb{R}^n$. A lattice is the set of all integer combinations of some linearly independent basis vector

$$\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \in \mathbb{R}^{n \times m} \colon \Lambda(\mathbf{B}) = \{\sum_{i=1}^{m} z_i \mathbf{b}_i, z_i \in \mathbb{Z}\}.$$

We consider n a power of two, such that the polynomial ring $\mathrm{R} = \mathbb{Z}[x]/(x^n + 1)$ is isomorphic to the integer lattice $\mathbb{Z}^n$. Then a polynomial $f = \sum_{i=0}^{n-1} f_i x^i$ in R corresponds to the integer vector of its coefficients $(f_0, \ldots, f_{n-1})$ in $\mathbb{Z}^n$. The notation norm of a polynomial $\|f\|$ means that we consider the norm of its coefficient vector, and as for the integer, the norm of a vector of polynomial $\|\mathbf{f}\| = \max_i \|f_i\|$. For the rest of this thesis we will work with polynomials over R, or $\mathrm{R}_q = \mathrm{R}/q\mathrm{R} = \mathbb{Z}_q[x]/(x^n + 1)$, where q is a prime verifying $q = 1 \pmod{2n}$.

Definition 1. *The dual lattice of a given lattice $\Lambda \subseteq \mathbb{R}^n$ is:*
$\Lambda^\star = \{\mathbf{x} \in \mathbb{R}^n \colon \forall\, \mathbf{y} \in \Lambda, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$.

We also define the minimum distance of a lattice denoted $\lambda_1(\Lambda)$:

Definition 2. *Let $\mathbf{v} \in \Lambda$ non-zero such that for every $\mathbf{x} \in \Lambda$ we have $\|\mathbf{x}\| \geqslant \|\mathbf{v}\|$ then $\lambda_1(\Lambda) = \|\mathbf{v}\|$, we have $\lambda_1(\Lambda) = \min_{\mathbf{v} \in \Lambda} \|\mathbf{v}\|$.*

We can also define the i-th minimal distance $\lambda_i(\Lambda)$ such that, given a set of i independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_i$ with $\lambda_1(\Lambda) = \|\mathbf{v}_1\|, \ldots, \lambda_{i-1}(\Lambda) = \|\mathbf{v}_{i-1}\|$ and $\mathbf{v}_i \in \Lambda$ being the shortest vector independent of $\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}$, then $\lambda_i(\Lambda) = \|\mathbf{v}_{i-1}\|$.

3.2.2 Problems on lattices

Worst-case problems. We first describe some problems that can be immediately linked to a given lattice Λ, with some basis $\mathbf{B}$ of Λ.

Definition 3 (Shortest Vector Problem)**.** *The SVP problem asks given an arbitrary basis $\mathbf{B}$ of a lattice Λ, to find the shortest non-zero vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v}\| = \lambda_1(\Lambda)$.*

This problem can be derivate on several variants that are either decisional or allow an approximation factor:

Definition 4 (Approximate Shortest Vector Problem)**.** *Given an approximate factor γ, the SVP_γ problem asks given an arbitrary basis $\mathbf{B}$ of a lattice Λ, to find a non-zero vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v}\| \leqslant \gamma \cdot \lambda_1(\Lambda)$.*

Definition 5 (Decisional Approximate Shortest Vector Problem)**.** *Given an approximate factor γ and an integer $d > 0$, the $GapSVP_\gamma$ problem asks given an arbitrary basis $\mathbf{B}$ of a lattice Λ, to distinguish between the two following cases:*

— $\lambda_1(\Lambda) \leqslant d$

— $\lambda_1(\Lambda) \geqslant \gamma \cdot d$.

Average-case problems. We now give some problems linked to a lattice Λ and an associated basis $\mathbf{A}$ that is randomly distributed in a uniform way. These problems are directly used to build the cryptographic primitives based on lattices and their security relies on the problems defined above.

Definition 6 ($\mathrm{SIS}_{n,q,\beta,m}$)**.** *Given a uniformly chosen matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find non-zero integer vector $\mathbf{s} \in \mathbb{Z}^m$ such that $\|\mathbf{s}\|_\infty \leq \beta$ and $\mathbf{A} \cdot \mathbf{s} = \mathbf{0} \mod q$. $\mathrm{SIS}_{n,q,\beta,m}$ is hard if for any adversary $\mathcal{A}$, if the probability to solve* SIS *is negligible, i.e. it is bounded by negl(λ). $\mathrm{SIS}_{n,q,\beta,m}$ is sub-exponentially hard if the probability to solve* SIS *is bounded by $2^{-O(n^\epsilon)} \cdot negl(\lambda)$ for some constant $0 < \epsilon < 1$.*

Definition 7 ($\mathrm{LWE}_{n,q,D_{\mathbb{Z},\alpha q},m}$)**.** *Given a uniformly chosen matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a vector $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \mod q$, with $\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,\alpha q}$. The search* LWE *problem asks to find $\mathbf{s}$, while the decisional* LWE *problem asks to distinguish if a $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ pair has been generated from the uniform distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ or if it has been generated as a LWE sample.*

Worst-case average-case reductions. The two above problems have been proven secure thanks to reductions called worst-case to average-case. Indeed the reduction starts from a lattice with a randomly distributed basis $\mathbf{A}$ and the reduction proves that solving SIS or LWE is as hard as solving a variant of the SVP problem in the worst case. Concerning SIS, for any $m = \mathrm{poly}(n), \beta > 0$ and $q \leqslant \beta \cdot \mathrm{poly}(n)$, then solving the $\mathrm{SIS}_{n,q,\beta,m}$ problem is at least as hard as solving the GapSVP_γ problem, with $\gamma = \beta \cdot \mathrm{poly}(n)$ [GPV08]. As for the LWE problem, let $m = \mathrm{poly}(n), q \leqslant 2^{\mathrm{poly}(n)}$ and $\alpha \cdot q \geqslant 2\sqrt{n}$, then solving the $\mathrm{LWE}_{n,q,D_{\mathbb{Z},\alpha q},m}$ is at least as hard as quantumly solve GapSVP_γ for an arbitrary lattice of dimension n and some $\gamma = \tilde{O}(n/\alpha)$ [Reg05].

Structured variants. In order to benefit a better efficiency, some structures are added into the lattices, the above problems are then reformulated according to these new structures:

We first define Ring-LWE that is similar to the LWE problem described above but on a polynomial ring:

Definition 8 (Ring-LWE$_{q,D_{\mathbb{R},\alpha q},m}$)**.** *Given a uniformly chosen polynomial vector* $\mathbf{a} \in \mathrm{R}_q^m$ *and a polynomial* $\mathbf{b} = \mathbf{a} \cdot s + \mathbf{e} \mod q$*, with* $s \leftarrow_{\$} \mathrm{R}_q$ *and* $\mathbf{e} \leftarrow D_{\mathrm{R}^m,\alpha q}$*. The search* LWE *problem asks to find* s*, while the decisional* LWE *problem asks to distinguish if a* $(\mathbf{a}, \mathbf{b}) \in \mathrm{R}_q^m \times \mathrm{R}_q^m$ *pair has been generated from the uniform distribution on* $\mathrm{R}_q^m \times \mathrm{R}_q^m$ *or if it has been generated as a LWE sample.*

We consider Ring-SIS, a variant of SIS, proven to be at least as hard as the SIVP problem on ideal lattices [LM06; PR06].

Definition 9 (Ring-SIS$_{q,m,\beta}$)**.** *Given* $\mathbf{a} = (a_1, \ldots, a_m)^T \in \mathrm{R}_q^m$ *a vector of* m *uniformly random polynomials, find a non-zero vector of small polynomials* $\mathbf{x} = (x_1, \ldots, x_m)^T \in \mathrm{R}^m$ *such that* $f_{\mathbf{a}}(\mathbf{x}) = \sum_{i=1}^m a_i.x_i = 0 \mod q$ *and* $0 < \|\mathbf{x}\| \leqslant \beta$*.*

In one of our scheme, we need an other variant called the k-SIS problem [BF11].

Definition 10 (Ring k-SIS, adapted from [BF11, definition 4.1])**.** *For any integer* $k \geqslant 0$*, an instance of the Ring k-SIS*$_{q,m,\beta,\sigma}$ *problem is a vector* $\mathbf{a} \in \mathrm{R}^m$ *and a set of* k *short polynomials vectors* $\mathbf{e}_1, \ldots \mathbf{e}_k \in \Lambda_q^{\perp}(\mathbf{a})$ *that are gaussians vector of width* σ*. A solution to the problem is a non zero polynomial* $\mathbf{v} \in \mathrm{R}^m$ *such that* $\|\mathbf{v}\| \leqslant \beta$*,* $f_{\mathbf{a}}(\mathbf{v}) = 0 \mod q$ *(i.e.,* $\mathbf{v} \in \Lambda_q^{\perp}(\mathbf{a})$*), and* $\mathbf{v} \notin \mathrm{R} - \mathrm{span}(\mathbf{e}_1, \ldots \mathbf{e}_k)$*.*

Hardness. Following the proof of [Lin+14] adapted to the ring setting, the hardness of Ring k-SIS$_{q,m,\beta,\sigma}$ is insured for $k = O(n)$ and $k < \dim(\ker(\mathbf{a}))$ by a reduction from a Ring-SIS$_{q,m,\beta'}$ problem with $\beta' = O(\beta \cdot k \cdot \mathrm{poly}(n))$.

3.2.3 Mathematical tools for lattices

Smoothing parameter. The smoothing parameter of a lattice Λ has been introduced by Micciancio and Regev [MR07]. The idea of this value is to determine what is the minimal value $\eta_\epsilon(\Lambda)$ such that the discrete gaussian distribution with a $\sigma \geqslant \eta_\epsilon$ parameter behaves like a continuous gaussian distribution.

Definition 11 ([MR07]). *For a lattice $\Lambda \subseteq \mathbb{R}^n$ and a positive real $\epsilon > 0$, he smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest real $\sigma > 0$ such that:*

$$\rho_{1/\sigma}(\Lambda^\star \setminus \{0\}) \leqslant \epsilon.$$

The vectors sampled from $D_{\Lambda,\sigma}$ are short with overwhelming probability.

Lemma 1 ([Ban93]). *For any lattice $\Lambda \subseteq \mathbb{R}^n, \sigma > \eta_\epsilon(\Lambda), \mathbf{c} \in \mathbb{R}^n$ for some $\epsilon \in (0,1)$, we have: $Pr_{\mathbf{x} \hookleftarrow D_{\Lambda,\sigma,\mathbf{c}}}[\|\mathbf{x}-\mathbf{c}\| \geqslant \sqrt{n}\sigma] \leqslant 2^{-n} \cdot \frac{1+\epsilon}{1-\epsilon}$.*

When sampling integers we have to tailcut the gaussian distribution. In order to do this, we use the fact that $Pr_{\mathbf{x} \hookleftarrow D_{\mathbb{Z},\sigma}}[|x| \leqslant t \cdot \sigma] \geqslant \text{erfc}(t/\sqrt{2})$, where $\text{erfc}(x) = 1 - 2/\pi \int_0^x \exp(-t^2)\,\mathrm{d}t$. In practice, for $\lambda = 100$ and $t = 12$, a vector $\mathbf{x} \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ will verify $\|\mathbf{x}\| \leqslant t \cdot \sigma \cdot \sqrt{n}$ with overwhelming probability.

We define the function Φ as $\Phi_{\sigma,\mathbf{c}}(\mathrm{B}) = Pr_{\mathbf{x} \hookleftarrow D_{\Lambda,\sigma,\mathbf{c}}}[\|\mathbf{x}-\mathbf{c}\| \leqslant \mathrm{B}]$, and we also define the following quantity $\bar{\rho}_{\sigma,\mathbf{c},\mathrm{B}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}}{\Phi_{\sigma,\mathbf{c}}(\mathrm{B})}$, $\forall \mathbf{x} \in \mathrm{S} \subset \mathbb{R}^n$ and $\bar{\rho}_{\sigma,\mathbf{c}}(\mathbf{x}) = 0$, $\forall \mathbf{x} \notin \mathrm{S}$, as a truncated gaussian. Finally we denote $\bar{D}_{\Lambda,\mathbf{c},\sigma,\mathrm{B}}$ as the truncated gaussian, where the elements $\mathbf{x} \in \Lambda$, such that $\|\mathbf{x}\| > \mathrm{B}$ have a probability density equals to 0.

We then define a lemma that argue about the statistical distance between two gaussian distributions.

Lemma 2 ([Gol+10, Lemma 3]). *Let $v \in \mathrm{R}$ be arbitrary. The statistical distance between the distributions $D_{\mathrm{R},\sigma}$ and $D_{\mathrm{R},\sigma,v}$ is at most $\frac{\|v\|}{\sigma}$.*

We finally give below the leftover hash lemma:

Lemma 3 (Leftover Hash Lemma [Hås+99]). *Let $m, n, q \geqslant 1$ be integers such that $m \geqslant 4n \log q$ and q prime. Let $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{r} \xleftarrow{\$} \{0,1\}^m$, then $(\mathbf{A}, \mathbf{Ar})$ is at negligible statistical distance from uniform distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^n$.*

3.3 Cryptographic tools

3.3.1 Encryption and signature protocols

Public key encryptions scheme. A public key encryption scheme is defined by the algorithms KeyGen, Encrypt and Decrypt that we develop below.

KeyGen**:** The KeyGen algorithm takes as input the security parameter λ and outputs a secret/public key pair (e_k, m_k), such that m_k is used to encrypt while e_k is used to decrypt.

Encrypt**:** The encryption algorithm takes as input the encryption key m_k and a message M to be encrypted, it outputs a ciphertext c.

Decrypt**:** The decryption algorithm takes as input the decryption key e_k and a ciphertext c, it outputs a message M or an error $\perp$.

Signature scheme. A signature scheme is defined by the algorithms KeyGen, Sign and Verify that we develop below.

KeyGen**:** The KeyGen algorithm takes as input the security parameter λ and outputs a secret/public key pair (s_k, p_k), such that p_k is used to verify the signature (and sign the message as well sometimes) while s_k is used to sign a message.

Sign**:** The signing algorithm takes as input the secret/public key (s_k, p_k) and a message M to be signed, it outputs a signature ν.

Verify**:** The decryption algorithm takes as input the verification key p_k and a signature ν, it outputs 1 if the signature is valid and 0 otherwise.

Security. The basic security asked for the encryption schemes is the *ciphertext IN-Distinguishability under Chosen-Plaintext Attack* (or IND-CPA), while the basic security asked for a signature scheme is the *Existential UnForgeability against Chosen-Message Attack* (or EUF-CMA), these two security are depicted below:

$\mathbf{Exp}_{\mathcal{D}}^{\text{IND-CPA}}(\lambda)$	$\mathbf{Exp}_{\mathcal{F}}^{\text{EUF-CMA}}(\lambda)$
$(e_k, m_k) \leftarrow \text{KeyGen}(1^\lambda)$	$(s_k, p_k) \leftarrow \text{KeyGen}(1^\lambda)$
$M_0, M_1 \leftarrow \mathcal{D}(m_k)$	$(M^\star, \nu^\star) \leftarrow \mathcal{F}^{\text{Sign}(s_k,\cdot)}(p_k)$
$b^\star \leftarrow_\$ \{0,1\}$	$d \leftarrow \text{Verify}(p_k, M^\star, \nu^\star)$
$c^\star \leftarrow \text{Encrypt}(m_k, M_{b^\star})$	We denote $(M_1, \nu_1), \ldots, (M_{qs}, \nu_{qs})$ the allowed queries of $\mathcal{F}$
$b \leftarrow \mathcal{D}(c^\star)$	to a Sign oracle.
Return 1 if $b = b^\star$	Return $d \cap (M^\star \neq M_i, \forall i \in [qs])$.

The (IND-CPA) property asks that an attacker, providing two messages M_0 and M_1

can not distinguish which one of them has been encrypted in $c^\star$. This property can be enforced into a (IND-CCA) property (for *ciphertext INDistinguishability under Chosen-Ciphertext Attack*) in which case he is allowed to make decryption queries on ciphertext of its choice different from the challenge $c^\star$.

Concerning the (EUF-CMA) property, an attacker is asked to output a valid signature on a message on which he has not already queried such a signature. This property can as well be enforced into a (SEUF-CMA) where the attacker is allowed to output a signature on a message already queried to a signing oracle, but obviously the signature output by the attacker must be different to the one output by the oracle.

Security model. It exists several security models that are considered in the various security proofs. We mention here the two main ones which are the random oracle model and the standard model that we make use in our works. The random oracle model (or ROM) has been introduced by Bellare and Rogaway [BR93] and allows every party to have access to a public random oracle, such that for every new input, it outputs a random string. In a real use case such an ideal function does not exist, but we often make the statement that a hash function is modeled as a random oracle. Concerning the standard model, it does not use any kind of idealization and is what is closest to what happens in practice.

3.3.2 Identifications scheme and Fiat-Shamir signatures

An identification scheme $\mathcal{ID}$ is composed of the following algorithms.

— **KeyGen**: The key generation algorithm takes as input the security parameter λ and outputs a key pair (pk, sk), such that the prover algorithm P gets the secret key sk and the verifier V algorithm the public key pk.

— **P**: The prover algorithm takes as input the secret key sk and the conversation transcript, it outputs the next message to be sent to the verifier.

— **V**: The verifier algorithm, that is first probabilistic and outputs a challenge CHL. At the end of the interactions it is deterministic, and takes as input the public key pk and the transcript (CMT,CHL,RSP) of the interaction and it outputs 1 if the transcript is valid and 0 otherwise.

From such an identification scheme, the Fiat-Shamir transformation [FS86], can turn it into a digital signature scheme. We describe below this transformation.

Fiat-Shamir transformation. Let $\mathcal{ID}$ be an identification scheme, and H:$\{0,1\}^\star \to \{0,1\}^c, c > 0$ be an hash function modeled as a random oracle. From this identification scheme $\mathcal{ID}$, we build a digital signature scheme $\mathcal{S} = (\text{KeyGen}, \text{Sign}, \text{Verify})$. This signature scheme is composed of the same KeyGen algorithm as the identification scheme, while the Sign and Verify algorithms are described below:

$\text{Sign}(sk, M)$	$\text{Verify}(pk, M, \sigma)$
$\text{CMT} \leftarrow \mathbf{P}(sk)$	parse σ as (CMT, RSP)
$\text{CHL} \leftarrow \text{H}(\text{CMT}, M)$	$\text{CHL} \leftarrow \text{H}(\text{CMT}, M)$
$\text{RSP} \leftarrow \mathbf{P}(sk, \text{CMT}, \text{CHL})$	Return $\mathbf{V}(pk, \text{CMT}, \text{CHL}, \text{RSP})$
Return $\sigma = (\text{CMT}, \text{RSP})$	

The digital signature built from this Fiat-Shamir transformation in the lattices, starts from the identification scheme of Lyubashevsky [Lyu08] and makes use of a technique called the rejection sampling. This technique introduced by Lyubashevsky in [Lyu08] and improved in [Lyu09; Lyu12], is used in the case we have a distribution depending on a secret we want to hide. The main idea is that the vector computed by the signer, in the signing step, is rejected or accepted following a probability that prevents the secret key to leak. The theorem below expresses this idea.

Theorem 1 ([Lyu12, theorem 4.6])**.** *Let V be a subset of $\mathbb{Z}^n$ in which all elements have norms less than T, σ be some element in $\mathbb{R}$ such that $\sigma = \omega(T\sqrt{\log(n)})$, and h:$V \to \mathbb{R}$ be a probability distribution. Then, there exists a constant $G = O(1)$ such that the distribution of the following algorithm $\mathcal{A}$:*

1:$\mathbf{v} \hookleftarrow_\$ h$ 2:$\mathbf{z} \leftarrow D_{\mathbb{Z}^n,\sigma,\mathbf{v}}$ 3: output $(\mathbf{z}, \mathbf{v})$ with probability $min(\frac{D_{\mathbb{Z}^n,\sigma}}{G.D_{\mathbb{Z}^n,\sigma,\mathbf{v}}}, 1)$

is within statistical distance $\frac{2^{-\omega(\log n)}}{G}$ of the distribution of the following algorithm $\mathcal{F}$:

1:$\mathbf{v} \hookleftarrow_\$ h$ 2:$\mathbf{z} \hookleftarrow_\$ D_{\mathbb{Z}^n,\sigma}$ 3: output $(\mathbf{z}, \mathbf{v})$ with probability $1/G$.

Moreover, the probability that $\mathcal{A}$ outputs something is at least $\frac{1-2^{-\omega(\log n)}}{G}$.

More concretely, if $\sigma = \delta T$ for any positive δ, then $G = \exp^{12/\delta + 1/(2\delta^2)}$, the output of the algorithm $\mathcal{A}$ is within statistical distance $\frac{2^{-100}}{G}$ of the output of $\mathcal{F}$, and the probability that $\mathcal{A}$ outputs something is at least $\frac{1-2^{-100}}{G}$.

We briefly describe below the Fiat-Shamir signature of Lyubashevsky [Lyu09] published in Asiacrypt 2009.

The KeyGen algorithm generates $\mathbf{S} \leftarrow_{\$} \{-d, \ldots, d\}^{m \times k}$ and $\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{n \times m}$ such that $\mathbf{T} = \mathbf{A} \cdot \mathbf{S} \mod q \in \mathbb{Z}_q^{n \times k}$ and an hash function H:$\{0,1\}^* \rightarrow \{\mathbf{v}\colon \mathbf{v} \in \{0,1\}^k, \|\mathbf{v}\|_1 \leqslant \kappa\}$, finally it outputs $(sk = \mathbf{S}, pk = (\mathbf{A}, \mathbf{T}, \mathrm{H}))$.

Sign(sk, M)

1.$\mathbf{y} \leftarrow D_{\mathbb{Z},\sigma}^m$

2.$\mathbf{c} \leftarrow \mathrm{H}(\mathbf{A} \cdot \mathbf{y}, M)$

3.$\mathbf{z} \leftarrow \mathbf{y} + \mathbf{S} \cdot \mathbf{c}$

5.Output $(\mathbf{y}, \mathbf{z})$ with probability $\min(\frac{D_{\mathbb{Z}^n,\sigma}}{G.D_{\mathbb{Z}^n,\sigma,\mathbf{S}\cdot\mathbf{c}}}, 1)$

Verify(pk, M, σ)

1.Parse σ as $(\mathbf{y}, \mathbf{z})$

2.$\mathbf{c} \leftarrow \mathrm{H}(\mathbf{A} \cdot \mathbf{y}, M)$

3.If $\mathrm{H}(\mathbf{A} \cdot \mathbf{z} - \mathbf{T} \cdot \mathbf{c}, M) = \mathbf{c}$ and $\|\mathbf{z}\| \geqslant \sigma\sqrt{m}$, output 1.

4.Else output 0.

The parameters are set such that the problem $\mathrm{SIS}_{n,q,\beta,m}$ is hard with $\beta = (2\sigma + 2d\kappa)\sqrt{m}$. Moreover the above scheme is proven to be EUF-CMA secure under the hardness of the SIS problem.

3.3.3 GPV-style signatures and trapdoors constructions

As introduced in [Ajt96] and widespread in [GPV08], a trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is a short basis of the lattice $\Lambda_q^{\perp}(\mathbf{A})$:= $\{\mathbf{v} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{v} = 0 \mod q\}$. A trapdoor allows to sample short Gaussian vectors solution of the ISIS problem: $\mathbf{A}\mathbf{v} = \mathbf{x} \mod q$ with $\mathbf{x} \in \mathbb{Z}_q^n$.

Trapdoors. For all $\mathbf{v} \in \mathbb{Z}_q^n$, $\mathbf{A}_{\gamma_0}^{-1}(\mathbf{v})$ is the random variable with discrete gaussian distribution $D_{\mathbb{Z}^m,\gamma_0}$ conditioned on $\mathbf{A} \cdot \mathbf{A}_{\gamma_0}^{-1}(\mathbf{v}) = \mathbf{v} \bmod q$. A γ_0-trapdoor for $\mathbf{A}$ allows a procedure that can sample from $\mathbf{A}_{\gamma_0}^{-1}(\mathbf{v})$ in time $\mathrm{poly}(n, m, \log q)$ for any $\mathbf{v} \in \mathbb{Z}_q^n$. By overloading notation we denote a γ_0-trapdoor for $\mathbf{A}$ by $\mathbf{A}_{\gamma_0}^{-1}$.

Lemma 4 (Trapdoor generation [Ajt96; MP12])**.** *There exists an efficient procedure, that we call* $\mathrm{TrapGen}(1^n, 1^m, q)$*, with an efficiently computable value* $m_0 = O(n \log q)$ *such that*

for all $m \geqslant m_0$ *outputs a pair* $(\mathbf{A}, \mathbf{A}_{\gamma_0}^{-1})$*, where* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *is at negligible distance from uniform and* $\mathbf{A}_{\gamma_0}^{-1}$ *is a* γ_0*-trapdoor for* $\mathbf{A}$ *with* $\gamma_0 = O(\sqrt{n \log q \log n})$*.*

We describe now the gadget-based trapdoor introduced in [MP12] in the ring $\mathbb{Z}_q[x]/(x^n + 1)$, that we use in some of our constructions in order to be more efficient.

Gadget-based trapdoor. In [MP12] trapdoors, the matrix $\mathbf{a} \in \mathrm{R}_q^m$ is constructed by picking a first part uniformly at random, and a second part almost uniformly at random by including a gadget structure, to help the search of a solution for a Ring-SIS problem. The trapdoor construction then uses a gadget vector $\mathbf{g} = (1, 2, 4, \ldots, 2^{k-1})^T \in \mathrm{R}_q^k$, with $k = \lceil \log_2 q \rceil$ for which the inversion of the function $f_{\mathbf{g}^T}(\mathbf{z}) = \mathbf{g}^T\mathbf{z} \in R_q$ is easy to compute.

Construction. The construction of the gadget-based trapdoor takes as input the modulus q, the Gaussian parameter τ, an optional $\mathbf{a}' \in \mathrm{R}_q^{m-k}$ and $h \in \mathrm{R}_q$. If no $\mathbf{a}'$ is given it is chosen uniformly in R_q^{m-k} and if no h is given, $h = 1$. The construction outputs a matrix $\mathbf{a} = (\mathbf{a}'^T \| h\mathbf{g} - \mathbf{a}'^T\mathbf{T})^T$ with $\mathbf{T} \in \mathrm{R}^{(m-k)\times k}$ its trapdoor associated to the tag h, generated as a Gaussian of parameter τ.

Preimage sampling. The preimage sampling, given $\mathbf{a} \in \mathrm{R}_q^m$, is the computation of a short vector solution $\mathbf{v} \in \mathrm{R}^m$ of a Ring-SIS problem $f_{\mathbf{a}}(\mathbf{v}) = \sum_{i=1}^m a_i.v_i = 0 \bmod q$, available only thanks to a trapdoor $\mathbf{T} \in \mathrm{R}_q^{(m-k)\times k}$ for $\mathbf{a}$. The construction of [MP12] enables the following algorithm PreSample($\mathbf{T}, x \in \mathrm{R}_q, \zeta$) for the preimage sampling $\mathbf{x} \in \mathrm{R}^m$, with width parameter ζ, of $f_{\mathbf{a}}(\mathbf{v}) = x$:

1. Find $\mathbf{z} \hookleftarrow D_{\mathrm{R},\alpha}^k$, satisfying $f_{\mathbf{g}}(\mathbf{z}) = h^{-1}(x - \mathbf{a}^T\mathbf{p})$, with $\mathbf{p} \in \mathbf{R}_q^m$ a perturbation vector with covariance matrix $\Sigma_{\mathbf{p}} = \zeta^2\mathbf{I}_m - \alpha^2\binom{\mathbf{I}_k}{\mathbf{T}}(\mathbf{T}^T\mathbf{I}_k)$.
2. Compute $\mathbf{v} = \mathbf{p} + \binom{\mathbf{I}_k}{\mathbf{T}}\mathbf{z}$ with covariance matrix $\Sigma_{\mathbf{v}} = \Sigma_{\mathbf{p}} + \alpha^2\binom{\mathbf{I}_k}{\mathbf{T}}(\mathbf{T}^T\mathbf{I}_k)$, satisfying $\mathbf{a}^T\mathbf{v} = \mathbf{a}^T\mathbf{p} + \mathbf{a}^T\binom{\mathbf{I}_k}{\mathbf{T}}\mathbf{z} = \mathbf{a}^T\mathbf{p} + h\mathbf{g}^T\mathbf{z} = \mathbf{a}^T\mathbf{p} + h.h^{-1}(x - \mathbf{a}^T\mathbf{p}) = x$.

GPV signature scheme. This type of trapdoor is used to build digital signature schemes, we describe above the construction of Gentry et al. [GPV08].

The KeyGen algorithm generates a pair $(\mathbf{A}, \mathbf{A}_{\gamma_0}^{-1})$ thanks to the above trapdoor generation algorithm. It outputs $(sk = \mathbf{A}_{\gamma_0}^{-1}, pk = \mathbf{A})$ as the secret/public key pair and selects a hash function H: $\{0,1\}^\star \to \mathbb{Z}_q^n$ modeled as a random oracle.

Sign(sk, M)

1.If it exists (M, σ_M) in a local storage, outputs (M, σ_M).

2.Else $\sigma_M \leftarrow \text{PreSample}(\mathbf{A}_{\gamma_0}^{-1}, \text{H}(M), \gamma)$ for $\gamma \geqslant \gamma_0$ and outputs σ_M.

Verify(pk, M, σ_M)

1.If $\mathbf{A} \cdot \sigma_M = \text{H}(M) \mod q$ and $0 < \|\sigma_M\| \leqslant \gamma\sqrt{m}$, outputs 1.

2.Else outputs 0.

The above scheme is proven to be EUF-CMA secure under the hardness of the SIS problem.

We describe now some additional tools on the trapdoors, allowing to delegate the trapdoors.

Lemma 5 (Trapdoor extension [ABB10; MP12])**.** *Let $\mathbf{M} \in \mathbb{Z}_q^{n\times m}$ be a matrix with trapdoor $\mathbf{M}_\gamma^{-1}$ and $\mathbf{N} \in \mathbb{Z}_q^{n\times p}$ a matrix such that $\mathbf{M} = \mathbf{NS}$ mod q where $\mathbf{S} \in \mathbb{Z}_q^{p\times m}$ with $s_1(\mathbf{S})$ its largest singular value. Then we can use $(\mathbf{M}_\gamma^{-1}, \mathbf{S})$ to sample from $\mathbf{N}_{\gamma'}^{-1}$ for any $\gamma' \geq \gamma \cdot s_1(\mathbf{S})$.*

Lemma 6 ([Cas+10, Lemma 3.2])**.** *There is a deterministic polynomial-time algorithm* ExtBasis *with the following properties: given an arbitrary $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ whose columns generate the entire group $\mathbb{Z}_q^n$, an arbitrary basis $\mathbf{S} \in \mathbb{Z}^{m\times m}$ of $\Lambda^\perp(\mathbf{A})$, and an arbitrary $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times\bar{m}}$,* ExtBasis$(\mathbf{S}, \mathbf{A}' = \mathbf{A}\|\bar{\mathbf{A}})$ *outputs a basis $\mathbf{S}'$ of $\Lambda^\perp(\mathbf{A}') \subseteq \mathbb{Z}^{m+\bar{m}}$ such that $\|\tilde{\mathbf{S}}'\| = \|\tilde{\mathbf{S}}\|$. Moreover the same holds even for any permutation of the columns of $\mathbf{A}'$.*

There exists a function RandBasis developed by [Cas+10], which verifies the following lemma:

Lemma 7 ([Cas+10, Lemma 3.3])**.** *Let $\mathbf{S}$ be a basis of a m-dimensional integer lattice Λ and a parameter $s \geq \|\tilde{\mathbf{S}}\| \cdot \omega(\sqrt{\log n})$. The algorithm* RandBasis$(\mathbf{S}, s)$ *outputs a new basis $\mathbf{S}'$ of Λ such that, with overwhelming probability, $\mathbf{S}'$ verifies $\|\mathbf{S}'\| \leqslant s \cdot \sqrt{m}$. Moreover, for any two basis $\mathbf{S}_0, \mathbf{S}_1$ of the same lattice and any $s \geqslant \max\{\|\tilde{\mathbf{S}}_0\|, \|\tilde{\mathbf{S}}_1\|\} \cdot \omega(\sqrt{\log n})$, the outputs of* RandBasis$(\mathbf{S}_0, s)$ *and* RandBasis$(\mathbf{S}_1, s)$ *are within* negl(n) *statistical distance.*

We further need an important property of lattice trapdoors ([ABB10],[MP12]):

Lemma 8. *For* $\mathbf{A} \in \mathbb{Z}_q^{n\times p}$ *and* $\mathbf{R} \in \mathbb{Z}_q^{p\times m}$ *with* $m = n\lceil \log q \rceil$*, one can compute* $[\mathbf{A}\|\mathbf{AR}+\mathbf{G}]_\gamma^{-1}$ *for* $\gamma = O(\sqrt{mp}\,\|\mathbf{R}\|_\infty)$.

3.3.4 Hash function and Forking Lemma

We make use of the hash function construction developed in [LM06]. Let R_q be a ring and $m \geqslant 1$ a positive integer. The hash function $h_{\mathbf{a}}$:$\mathrm{R}_q^m \to \mathrm{R}_q$ for $\mathbf{a} \in \mathrm{R}_q^m$ is defined as: $\mathbf{x} \mapsto \langle \mathbf{a}, \mathbf{x}\rangle = \sum_{i=0}^{m-1} a_i x_i$. This hash function family will be denoted $\mathcal{H}(\mathrm{R}_q, m)$. We define the collision problem associated as follows.

Definition 12 (inspired by [Rüc10, definition 2.1])**.** *Let* $D \subset \mathrm{R}$*, the collision problem* $Col(\mathcal{H}(\mathrm{R}_q, m), D)$ *asks to find a distinct pair* $(\mathbf{x}, \mathbf{x}') \in D^m \times D^m$ *such that* $h(\mathbf{x}) = h(\mathbf{x}')$ *for* $h \leftarrow \mathcal{H}(\mathrm{R}_q, m)$.

Lemma 9 ([BN06, Lemma 1])**.** *Fix an integer* $q \geqslant 1$ *and a set* H *of size* $h \geqslant 2$*. Let* $\mathcal{A}$ *be a randomized algorithm that on input* $x, h_1, \ldots, h_q$ *returns a pair, the first element of which is an integer in the range* $0, \ldots, q$ *and the second element of which we refer to as a side output. Let* $\mathcal{IG}$ *be a randomized algorithm that we call the input generator. The accepting probability of* $\mathcal{A}$*, denoted* acc*, is defined as the probability that* $J \geqslant 1$ *in the experiment:*
$x \leftarrow_\$ \mathcal{IG}; h_1, \ldots, h_q \leftarrow_\$ H; (J, \sigma) \leftarrow_\$ \mathcal{A}(x, h_1, \ldots, h_q)$.
The forking algorithm $\mathcal{F}_\mathcal{A}$ *is the randomized algorithm that takes input* x *proceeds as follows:*

Algorithm $\mathcal{F}_\mathcal{A}(x)$

Pick coins ρ *for* $\mathcal{A}$ *at random*
$h_1, \ldots, h_q \leftarrow H$
$(I, \sigma) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q; \rho)$
If $I = 0$ *then returns* $(0, \epsilon, \epsilon)$
$h'_I, \ldots, h'_q \leftarrow H$
$(I', \sigma') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q; \rho)$
If $(I = I'$ *and* $h_I \neq h'_I)$ *then return* $(1, \sigma, \sigma')$
Else return $(0, \epsilon, \epsilon)$.

Let $\mathrm{frk} = \Pr[b = 1 : x \leftarrow \mathcal{IG}; (b, \sigma, \sigma') \leftarrow \mathcal{F}_\mathcal{A}(x)]$*. Then:* $\mathrm{frk} \geqslant \mathrm{acc}.(\frac{\mathrm{acc}}{q} - \frac{1}{h})$*. Alternatively,* $\mathrm{acc} \leqslant \frac{q}{h} + \sqrt{q.\mathrm{frk}}$.

3.3.5 Secret Key Encryption (SKE) from LWE.

We present now the secret key encryption scheme from LWE defined by [KY19]. The SKE scheme described below is IND-CPA secure and has a decryption circuit with $O(\log \lambda)$-depth. We need IND-CCA security which can be achieved from IND-CPA security and message authentication codes (MAC) scheme by a generic construction as explained in [KY19].

The SKE scheme below is a secret key variant of the Regev's scheme [Reg05]. The parameters of the scheme are the following: λ is the security parameter, l is the dimension of the message space $\mathcal{M} = \{0,1\}^l$, $n = \text{poly}(\lambda)$, q is a prime polynomially bounded with $q \geq 24n + 2$ and $m = \lambda + n\lceil \log q \rceil$

$\mathsf{SKE.Setup}(1^\lambda)$ Set the dimensions n and m of the matrix and the modulus q. Outputs $\mathsf{pp} = (n, m, q)$.

$\mathsf{SKE.Gen}(\mathsf{pp})$ Sample two matrices $\mathbf{S}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $\mathbf{S}_1 \xleftarrow{\$} \mathbb{Z}_q^{n\times l}$. Output the secret key $K = (\mathbf{S}_0, \mathbf{S}_1)$

$\mathsf{SKE.Enc}(K, M)$ Parse K as $(\mathbf{S}_0, \mathbf{S}_1)$. Sample $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n \setminus \{\mathbf{0}\}, \mathbf{x}_0 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^m, 3\sqrt{n}}$ and $\mathbf{x}_1 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^l, 3\sqrt{n}}$. Compute $\mathbf{c}_0^\top := \mathbf{a}^\top\mathbf{S}_0 + \mathbf{x}_0^\top \quad \mathbf{c}_1^\top := \mathbf{a}^\top\mathbf{S}_1 + \mathbf{x}_1^\top + \lceil q/2 \rceil \cdot M$, where $M \in \{0,1\}^l$ is considered as a row vector. Finally, output the ciphertext $\mathsf{ct} := (\mathbf{a}, \mathbf{c}_0, \mathbf{c}_1)$

$\mathsf{SKE.Dec}(K, \mathsf{ct})$ Parse ciphertext as $\mathsf{ct} = (\mathbf{a}, \mathbf{c}_0, \mathbf{c}_1)$ and check its validity.
Compute $\mathbf{v}_0^\top := \mathbf{c}_0^\top - \mathbf{a}^\top\mathbf{S}_0$ and $\mathbf{v}_1^\top := \mathbf{c}_1^\top - \mathbf{a}^\top\mathbf{S}_1$. If $\mathbf{v}_0 \notin [-3n, 3n]^m$, output $\mathsf{Invalid}$. Otherwise, recover $M_i \in \{0,1\}^l$ for $i \in [l]$ as follows: if the i-th coefficient of $\mathbf{v}_1$ is in $[-3n, 3n]$ then M_i is 0, otherwise, M_i is 1.

Correctness. This follows since we have $\|\mathbf{x}_0\|_\infty$, $\|\mathbf{x}_1\|_\infty \leq 3n$ with probability 1 and $q \geq 24n + 2$.

3.3.6 One-Time Signature (OTS) scheme.

A one-time signature scheme is defined by the following algorithms:

$\mathsf{OTS.KeyGen}(1^\lambda)$ is a randomized algorithm taking as input a security parameter 1^λ and outputs a verification key ovk and a signing key osk.

$\mathsf{OTS.Sign}(\mathsf{osk}, M)$ takes as input a secret key osk and a message M and outputs a signature σ.

$\mathsf{OTS.Verify}(\mathsf{ovk}, \sigma, M)$ takes as input a verification public key ovk, a message M and a signature σ and outputs Valid or $\mathsf{Invalid}$.

Correctness. For all λ, $(\mathsf{ovk}, \mathsf{osk}) \in \mathsf{OTS.KeyGen}(1^\lambda)$, M in the message space and $\sigma \in \mathsf{OTS.Sign}(\mathsf{osk}, \mathsf{M})$, $\mathsf{OTS.Verify}(\mathsf{ovk}, \sigma, M) = \mathsf{Valid}$ holds.

The security notion required for OTS scheme is the classical *strong unforgeability* with the difference that the adversary is allowed to make a single signing query instead of polynomially many. We define it using a game model between a challenger and an attacker $\mathcal{A}$ to define this security notion.

Setup: At the beginning of the game, the challenger runs $\mathsf{OTS.KeyGen}(1^{\lambda}) \rightarrow (\mathsf{ovk}, \mathsf{osk})$ and gives 1^λ and ovk to $\mathcal{A}$.

Signing Query: During the game $\mathcal{A}$ can perform a single signing query. When receiving M, the challenger runs $\mathsf{OTS.Sign}(\mathsf{osk}, M) \rightarrow \sigma$ and returns (M, σ) to $\mathcal{A}$.

Forgery: Eventually $\mathcal{A}$ outputs (M^*, σ^*) as a forgery. Then $\mathcal{A}$ wins the game if $\mathsf{OTS.Verify}(\mathsf{ovk}, \sigma^*, M^*) = \mathsf{Valid}$ and $(M, \sigma) \neq (M^*, \sigma^*)$

We then define the advantage of the attacker $\mathcal{A}$ as the probability that $\mathcal{A}$ wins the above game.

We say that an OTS scheme is strongly unforgeable, if for all PPT adversary, the advantage in the above game is negligible. .

Chapter 4

Forward Secure Group Signature in the standard Model

We present in this chapter a forward-secure lattice-based group signature which is secure in the standard model, and has a size of signature constant in the number of group members. Our construction follows an intensive line of research on the subject of lattice-based group signatures started in 2010 by Gordon et al. [GKV10]. The former construction and most of the existing schemes are restrained by the lack of efficiency of lattice-based zero-knowledge proofs of knowledge at that time, since the efficiency of the generic framework [BMW03] mainly relies on such constructions. This chapter is based on the following publication:

— [CGKRT20] Sébastien Canard, Adela Georgescu, Guillaume Kaim, Adeline Roux-Langlois and Jacques Traoré. Constant-size lattice-based group signature with forward secrecy in the standard model. Published in *PROVSEC 2020*.

We begin by giving the definition of a group signature and what security properties, a group signature must or can fulfill. Next, we give a short summary of the history of group signatures with a focus on the lattice-based constructions. We then present in detail, two recent constructions that we use to build our scheme: the group signature of [KY19] and the attribute-based signature of [Tsa17], that are both secure in the standard model. We develop as well the transformation of [Lin+19] that we apply on these schemes to get the property of forward security that we are looking for. We end the chapter with the description of our group signature, by presenting the different algorithms that compose the construction and we finally prove the security of our scheme.

4.1 Definitions and constructions

Group signatures were introduced as a new type of signatures by Chaum and van Heyst [CH91] in 1991 with the aim to rise in a context of privacy preserving cryptography. Indeed, they allow dedicated members of a group to sign messages while the identity of the signer remains hidden from the point of view of the verifier (anonymity). The latter can only ensure that a member, who belongs to a given group, has signed the message. Nevertheless, if necessary, the signature can be opened by an entity called group manager who holds some secret information and has the ability to reveal the identity of the signer (traceability). These features make group signatures very useful for real life applications including e-commerce systems, anonymous online communications and trusted hardware attestations.

4.1.1 Generic definition and basic properties

Considering a group of persons, a group signature verifies the following properties:

1. Only the members of the group can sign a message on behalf of this given group,
2. Anyone can check that someone who produced a signature actually belongs to the group,
3. In case of dispute, a manager can open a signature and reveal the identity of the group member who has generated it.

Concretely, to prove its membership, it means that every group member holds a certificate (which can take the form of a secret key, or a signature for example). The difficulty then, is to anonymously prove that he owns such a certificate in a way that, in case of dispute, an authority manages to open the signature and recovers the identity of the group member who generated it.

Following the former paper of [CH91], a lot of constructions were proposed, for example [Cam97; Pet97; AT99; Ate+00; BBS04], relying on classical assumptions such as the factorization problem or the discrete logarithm one. Regarding the security properties asked by such a primitive, for a long time there was no formal definition of security for group signatures. Besides of the anonymity and traceability properties introduced with the former paper of Chaum and van Heyst, some additional properties were defined such as:

— Unlinkability: no one, but the opening manager, can tell that two group signatures have been generated by the same group signer.

— Unforgeability: without the knowledge of a secret key, it is not possible to generate a valid group signature.

— Collusion resistance: it is not possible for any subset of group members colluding together, to output a valid signature that do not open on the identity of one member of the collusion.

— Exculpability: any group member (even the group manager) cannot produce a group signature that opens on the identity of another group member.

All these properties have been introduced in the subsequent papers, but without formal definitions and discussions on how they interact from each other.

In 2003, Bellare, Micciancio and Warinschi [BMW03] formalized the security requirements we can expect from a group signature scheme. More particularly, they gathered the properties mentioned previously into (full-)anonymity and (full-)traceability security, which encompassed every other security properties introduced in previous papers. Moreover, as in classical digital signature, the property of correctness is required. It demands that a group signature that has been genuinely generated, passes the verification test with, at least, an overwhelming probability.

Concerning the constructions, even if there exists several possible frameworks to build a group signature, one has known a lot of success and is at the starting point of the most efficient constructions. It is known as the "sign - then encrypt - then prove "framework, which has been developed by Bellare et al. [BMW03], who were inspired from previous works (e.g [Ate+00]). They based their constructions on the idea to give a certificate of membership to every group member, allowing them to prove their ownership in an anonymous way. This type of construction needs several building blocks, the first one is a public key encryption scheme that is IND-CCA secure, the second one is an adaptive simulation-sound NIZK proof for NP, finally the last primitive that is needed is a digital signature scheme secure against chosen-message attacks.

Combining all these primitives, the idea of the construction of the group signature is the following:

— In order to join the group, a member generates its own secret/public key pair and authenticates it to the group manager.

— Then the signer encrypts its certificate, composed of the key pair that has been

authenticated by the group manager, together with its identity and the message, finally it generates a zero-knowledge proof that all these elements are valid and well-formed.

— To check the validity of the group signature, a verifier just needs to verify that the zero-knowledge proofs are correct and include the elements claimed by the signer.

— In case of dispute, the opening manager, who owns the secret key of the encryption scheme, can open any signature to retrieve the identity of the group member that has output a given signature.

Group signatures, as defined above, are sometimes too limited for real use cases, and some additional features can be brought on top of this classical definition. For example, without any other mention, a group signature scheme is considered static. It means that once the secret key of each member is generated and given to its corresponding group member, no one can leave or join the group. In some scenarios this property narrows the use case in which we want to use the group signature. To soften this limitation, we can add some dynamic flavor. The most permissive is the fully dynamic property, giving the ability to any group member to join or leave the group at any moment, but there is also some schemes with mitigated dynamic properties, allowing member only to leave or join the group (partial dynamicity), sometimes using tools (such as verifier-local revocation).

Besides the dynamic problematic, another nice property which interests us, is the forward security. Coming back to the static group signatures, consider the case where a secret key of a group member leaks, then all the previous group signatures have to be invalidated and the whole group has to be generated again. To circumvent such a disaster, we can split the time into periods. Given all the time periods, we can then imagine a mechanism where all the secret keys of the group member are updated at the end of each time period. By this mean, if a secret key is stolen, we just have to discard this specific secret key for the current and next periods, instead of canceling everything.

Coming back to the group signatures on classical assumptions, we denote that the most effective group signatures to date, are those built on pairings, coming from the work of [BBS04; CL04] to most recent ones [Cam+20; CS18]. This efficiency is even more critical concerning the schemes that are built on lattices, since the NIZK proofs efficiency is really not closed compared to those on pairings. This problem induced a lot of research works for having an efficient, or equipped with nice properties, group signature on lattices.

4.1.2 Classical constructions and history of lattice-based group signatures

Group signatures have been the witnesses of an heavy history on lattices. From 2010 until today, at least 15 group signature schemes were born, with the question of finding an efficient lattice-based group signature still being an open question. In fact the NIZK proofs must be suitable with the others cryptographic primitives such as the encryption or signature schemes. This problematic, that is important on lattices, made the subject of obtaining an attractive post-quantum group signature on lattices, a challenging and interesting problem to solve.

For a long time the problem of inefficiency of the lattice-based group signature concerned the size of the output signature and/or public key. This is why, we make the convention that when we talk about linear size or logarithmic size, we mean size depending on the number of group members.

Coming back to 2010, Gordon et al. [Gol+10] proposed the first construction of a lattice-based group signature. Their construction encounters several limitations, such as a signature size and public key size that are linear. This scheme was quickly followed by a scheme from Camenisch et al. in 2012 who succeeds to lower the public key size from linear to logarithmic, but remains with a signature size which is linear. Both of these schemes used non-interactive zero-knowledge proof of [MV03] which limits the signature size to linear.

The first scheme that enjoys a logarithmic size both in the public key size and in the signature size was designed by Laguillaumie et al. [Lag+13] and relies on a new non-interactive zero-knowledge proof obtained by transforming the identification scheme of [Lyu08] into a non-interactive zero-knowledge proof using the Fiat-Shamir transformation [FS86]. Following this work, a lot of constructions in the logarithmic size [NZZ15] were designed, achieving verifier-local revocation property [Lan+14], using non-interactive zero-knowledge proofs *à la Stern* [Lin+13] to build their scheme [LNW15], using accumulators [BM93] to get static group signature [Lib+16b] and fully dynamic group signature [Lin+17], beneficing partial dynamicity [Lib+16a] or the property of message dependent opening [LMN16]. Finally in 2018, Ling et al. [San+18] developed the first lattice-based group signature with a signature size and a public key size independent of the number of member.

In complement of all these works, at Eurocrypt 2019, Katsumata and Yamada [KY19]

designed a lattice-based group signature departing from the usual paradigm of [BMW03] by replacing the usual non-interactive zero-knowledge proof by an attribute-based signature, trying by this mean to get rid of the limitations imposed by the problem of NIZK proofs on lattices. It is on this new model that we base our work and try to improve and add some properties to their construction. We first explain in the next part what tools are needed for this new model, then we develop the framework in detail before presenting our own work.

4.1.3 Some useful tools for standard model construction

Since the framework of Katsumata and Yamada [KY19] does not make any use of non-interactive zero-knowledge proofs, it has to replace them with other primitives. But first, we need to introduce several tools that are needed in their construction.

The first of these tools is a hash function family, called admissible hash functions. These hash functions are useful in scenarios in which we want to split the message space into two subsets, a controlled subset and a challenge subset. Concretely in simulation-based proofs, the simulator can then answer to queries asked by the attacker with the first subset, while the second subset is expected to contain the attacker output, in order for the simulator to complete the proof by solving a hard problem.

Admissible hash functions. Intuitively, WldCmp is a string comparison function with wildcards which takes as input three strings y, z, w and compares z and w only at those points where $y_i = 1$.

Definition 13 ([KY19, definition 1])**.** *Let $\ell := \ell(\lambda)$ and $\ell' := \ell'(\lambda)$ be some polynomials. We define the function* WldCmp*:* $\{0,1\}^{\ell} \times \{0,1\}^{\ell} \times \{0,1\}^{\ell} \to \{0,1\}$ *as*

$$\mathsf{WldCmp}(y, z, w) = 0 \Leftrightarrow \forall i \in [\ell], ((y_i = 0) \vee (z_i = w_i)),$$

where y_i, z_i and w_i denote the i-th bit of y, z and w respectively. Let $\{\mathrm{H}_\lambda : \{0,1\}^{\ell'(\lambda)} \to \{0,1\}^{\ell(\lambda)}\}_{\lambda \in \mathbb{N}}$ *be a family of admissible hash functions if there exists an efficient algorithm* AdmSmp *that takes as input 1^λ and $Q \in \mathbb{N}$ and outputs $(y, z) \in \{0,1\}^{\ell} \times \{0,1\}^{\ell}$ such that for every polynomial $Q(\lambda)$ and all $X^*, X^{(1)}, \cdots, X^{(Q)} \in \{0,1\}^{\ell'(\lambda)}$ with $X^* \notin \{X^{(1)}, \cdots, X^{(Q)}\}$, we have:*

$$\Pr_{(y,z)}\left[\mathsf{WldCmp}(y, z, \mathrm{H}(X^*)) = 0 \ \wedge \ (\mathsf{WldCmp}(y, z, \mathrm{H}(X^{(j)})) = 1 \ \ \forall j \in [Q])\right] \geqslant \Delta_Q(\lambda),$$

for a noticeable function $\Delta_Q(\lambda)$*, where the probability above is taken over the choice of* $(y, z) \xleftarrow{\$} \mathrm{AdmSmp}(1^\lambda, Q)$.

Besides of the above hash function family, the framework of [KY19] makes also use of evaluation functions.

Evaluation functions. In order to generate or check the validity of a signature, we need to execute some evaluation of a function with a set of lattices as input. The output of this evaluation is 1 if the function evaluated on an attribute x is not valid and 0 if the evaluation is correct. We use the notations and definition of the evaluation functions developed by Tsabary [Tsa17]. Moreover we denote $[x_1\mathbf{G}|\cdots|x_\ell\mathbf{G}]$ by $\mathbf{x} \otimes \mathbf{G}$ with $\mathbf{x} = (x_1, \cdots, x_\ell) \in \{0,1\}^\ell$.

Theorem 2 ([Tsa17, Theorem 2.7])**.** *There exist efficient deterministic algorithms* EvalF *and* EvalFX *such that for all* $n, q, \ell \in \mathbb{N}$*,* $m = n\lceil \log q \rceil$*, and for any sequence of matrices* $\overrightarrow{\mathbf{A}} = (\mathbf{A}_1, \cdots, \mathbf{A}_l) \in (\mathbb{Z}_q^{n\times m})^\ell$*, for any depth* d *boolean circuit* f*:*$\{0,1\}^\ell \to \{0,1\}$ *and for every* $\mathbf{x} = (x_1, \cdots, x_\ell) \in \{0,1\}^\ell$*, the outputs* $\mathbf{H}_f = \mathsf{EvalF}(f, \overrightarrow{\mathbf{A}})$*, and* $\mathbf{H}_{f,\mathbf{x}} = \mathsf{EvalFX}(f, \mathbf{x}, \overrightarrow{\mathbf{A}})$ *are in* $\mathbb{Z}^{(\ell m)\times m}$ *and it holds that* $\|\mathbf{H}_f\|_\infty, \|\mathbf{H}_{f,\mathbf{x}}\|_\infty \leqslant (2m)^d$ *and* $(\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{f,\mathbf{x}} = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_f - f(\mathbf{x})\mathbf{G} \mod q$.

4.1.4 Group signature in the standard model, attribute-based signature and forward secure framework

We now describe three existing constructions, that are the building blocks of our scheme: the group signature scheme of [KY19], the ABS proposed by Tsabary [Tsa17], and finally, the forward-secure mechanism of the group signature construction of [Lin+19].

Group Signature Scheme without NIZK. The starting point of our work is the recent lattice-based group signature scheme without NIZK in the standard model [KY19]. As stated previously, prior to the construction of [KY19], all works on group signatures on lattices were relying on the Sign-Encrypt-Prove framework defined by Bellare, Micciancio and Warinschi [BMW03]. In this framework, to sign a message M, a user encrypts both his certificate received from the group manager and a digital signature on M. Finally, he proves in a non-interactive zero-knowledge way that every element is well formed. Until recently (2019), constructing NIZK from lattices for any NP language was a long-standing open problem and by that time Katsumata and Yamada [KY19] proposed a

group signature scheme that by-passed the utilization of NIZK by replacing it with *indexed* attribute-based signature scheme (ABS). Their idea is based on the fact that for group signatures the needed NIZK is in the common reference string (CRS) model and, in the context of group signatures, it resembles to *designated-prover* NIZK (DP-NIZK) where there is a proving key k_P that needs to be kept secret (and thus is not known to the verifier, assuring zero-knowledge) and a verification key k_V which is public. Anyway, simply replacing NIZK in the CRS model with DP-NIZK is not enough since it trivially breaks anonymity. The breakthrough idea of Katsumata and Yamada was to view ABS as DP-NIZK. In attribute-based signatures, a signer with an attribute x is provided a secret key sk_x from the authority and can anonymously sign a message associated with a policy C using his secret key, if and only if, his attribute satisfies the policy C. In particular, the signature hides the attribute (anonymity) and users can not collude to pull their attributes together if none of the attributes satisfies the policy associated to the message (unforgeability). Now, an ABS can be seen as a DP-NIZK by the following association: the attribute x is seen as a witness w and the ABS signing key sk_x can be set as the proving key k_P of the DP-NIZK. Thus proving that w is a valid witness to a statement s i.e. $(s, w) \in \mathcal{R}$ for the NP relation $\mathcal{R}$ resorts to, firstly prepare a circuit $C_s(w) = \mathcal{R}(s, w)$ that has the statement s hard-wired into it, secondly sign a message associated with the policy C_s using the proving key $k_P = sk_x$ and finally output the signature as the NIZK proof π. Anonymity and unforgeability of the ABS assure the zero-knowledge property and soundness respectively.

Having shown a way of substituting the NIZK with ABS, it remains to indicate how to use ABS to construct group signature. We briefly explain, in the following, the general framework from [KY19]. The group manager issues for user i a key K_i of a secret key encryption (SKE) scheme and an ABS signing key $sk_{i||K_i}$ where $i||K_i$ is seen as an attribute. To sign a message M, the group member i encrypts his identity under K_i obtaining $\mathsf{ct}_i = \mathsf{SKE.Enc}(K_i, i)$ and creates an attribute-based signature for some policy C_{ct_i} which serves as a NIZK proof of the fact that ct_i encrypts the identity. The circuit C_{ct_i} has the statement ct_i hardwired such that $C_{\mathsf{ct}_i}(i||K_i) := (i = \mathsf{SKE.Dec}(K_i, \mathsf{ct}_i))$. The traceability property of the group signature holds from unforgeability of ABS and anonymity holds from anonymity of the ABS and semantic security of the SKE.

As for the instantiation of the ABS from lattices, [KY19] gives two possible solutions: the first one uses the ABS proposed by Tsabary [Tsa17] proven secure under the SIS assumption and the second one is an indexed ABS designed by them, relying also on

the SIS assumption. The need for the second construction is explained by the problems encountered when trying to plug the first construction into a group signature. Tsabary's scheme achieves selective unforgeability which is not enough for the security purposes of group signatures. Adaptiveness is the required property and can be easily achieved via complexity leveraging with the drawback that this approach requires a subexponential security loss. The two different ABS constructions give two different group signature schemes with the following properties:

(i) Tsabary's ABS gives rise to a group signature scheme with public key and signature size constant (independent) in the number of users and whose security relies on the hardness of LWE with polynomial approximation factor and subexponential hardness of SIS with polynomial approximation factor.

(ii) The second ABS gives rise to a group signature scheme with public key and signature size linear in the number of users whose security relies on the hardness of LWE and SIS with polynomial approximation factors.

Attributed Based Signature from Constrained Signature of Tsabary. The main building block of our group signature is an Attribute Based Signature scheme. In the following we briefly explain the ABS developped in Tsabary's paper. Then we show how to extend it to forward-secure ABS and use it as a building block for our signature scheme. First of all, the construction in Tsabary's paper is not really an *attribute-based signature* but rather a *key-policy constrained signature* or simply *constrained signature*. We note that the other flavour of constrained signatures, as defined in [Tsa17], called *message-policy constrained signature* is equivalent to attribute-based signatures. In constrained signatures, a signing key sk_f is associated with a policy $f{:}\{0,1\}^* \to \{0,1\}$, called the constraint, and a key sk_f can sign a message $x \in \{0,1\}^*$ only if the message satisfies the policy i.e. $f(x) = 0$. In attribute-based signatures each key is associated with an attribute $x \in \{0,1\}^*$ and a key sk_x can sign a policy f only if the attribute satisfies the policy i.e. $f(x) = 0$. A constrained signature can be easily transformed into an attribute-based signature using universal circuits (which we denote U_x) as briefly explained in [KY19] (but not done there), transformation that we sketch below.

The ABS scheme (as well as the original constrained signature of [Tsa17]) is built from lattice trapdoors. The verification key vk consists of a uniformly sampled matrix vector $\overrightarrow{\mathbf{A}} = [\mathbf{A}_1\|...\|\mathbf{A}_\ell] \in \mathbb{Z}_q^{n\times(m\times\ell)}$ (with ℓ the input size of the circuit C) and a close to uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times p}$ while the master signing key msk is a trapdoor for $\mathbf{A}$ denoted

$\mathbf{A}_{\gamma_0}^{-1}$. The signing key sk_{x_i} is associated to an user i (we prefer the simplified version of this notation even though it would be clearer to use $sk_{U_{x_i}}$ as notation) and to an universal circuit U_{x_i} (which has the attribute hard-wired and takes as input the policy (circuit) and a message). The secret key sk_{x_i} is a trapdoor $[\mathbf{A}\|\mathbf{A}_{x_i}]_{\gamma}^{-1}$ where $\mathbf{A}_{x_i} = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_{x_i} \in \mathbb{Z}_q^{n\times m}$ is computed from $\overrightarrow{\mathbf{A}}$ and U_{x_i} using the function EvalF. This function, associated with a function EvalFX, allows to compute $\mathbf{H}_{x_i} = \mathsf{EvalF}(x_i, \overrightarrow{\mathbf{A}})$, and $\mathbf{H}_{x_i,\mathbf{x}} = \mathsf{EvalFX}(x_i, \mathbf{x}, \overrightarrow{\mathbf{A}})$ both in $\mathbb{Z}^{(\ell m)\times m}$ and of bounded norm such that $(\overrightarrow{\mathbf{A}} - \mathbf{x}\otimes\mathbf{G})\cdot\mathbf{H}_{x_i,\mathbf{x}} = \overrightarrow{\mathbf{A}}\cdot\mathbf{H}_{x_i} - U_{x_i}(\mathbf{x})\mathbf{G}$ mod q, where $\mathbf{G}$ is the gadget matrix. Then, the manager can easily generate the secret key sk_{x_i} using it's own trapdoor $\mathbf{A}_{\gamma_0}^{-1}$. A valid signature for a message M, a circuit C and an attribute x_i is a short vector σ such that $[\mathbf{A}\|\overrightarrow{\mathbf{A}} - x_i \otimes \mathbf{G}]\cdot\sigma = \mathbf{0} \mod q$. We note that for every tuple (C, M, x_i), a trapdoor $[\mathbf{A}\|\overrightarrow{\mathbf{A}} - x_i\otimes\mathbf{G}]_{\gamma'}^{-1}$ can be derived from $[\mathbf{A}\|\overrightarrow{\mathbf{A}} - U_{x_i}(C,M)\mathbf{G}]_{\gamma'}^{-1}$ when $U_{x_i}(C,M) = C(x_i) = 0$.

We remark that, at this stage, the unforgeability of the ABS can easily be broken, as explained in [KY19] because the message is not bounded to the signature (both signature and verification just ignore the message) and a valid signature for a pair of policy and message (C, M) is also valid for (C, M') for $M \neq M'$. Therefore, in the security game, we can not allow signature queries and following the idea of [KY19], we use the fact that a scheme that is unforgeable only when the adversary can not make signature queries can be generically transformed into a scheme that is unforgeable even when the adversary is allowed to make signature queries. In short, the idea in [KY19] is to answer the signing queries using the secret key of a dummy user which does not exist in the real system. We will need to partition the set of all possible message-policy pairs into a challenge set and a controlled set (using admissible hash functions) with the hope that the adversary asks queries that fall into the controlled set to which the challenger can answer with the help of the dummy key. We also hope that the attacker outputs a forgery in the challenge set to allow the simulator to solve a hard problem.

Forward Secure Group Signature of [Lin+19]. Recall that to achieve forward-secure group signature, one needs a one-way key evolving mechanism for deriving secret keys for every period of time. Let us now briefly explain this mechanism following the idea of [Lin+19]. Let $T = 2^d$ be the total number of time periods, the time periods are represented in a binary tree, where each time period is a leaf of the tree. Each user secret key for a time period t is then associated with a sub-tree of depth d which uniquely defines the time period t. Let z be a binary string (corresponding to a time period) of

lenth d_z. The set $\text{Nodes}_{(t \leftarrow T-1)}$ contains nodes for which bases (trapdoors) are derived at a current period of time t and which allow to compute subsequent keys in the key update algorithm using the bonsai tree technique [Cas+10]. Each user will have associated a matrix corresponding to period time $z \in \text{Nodes}_{(t \leftarrow T-1)}$: $\mathbf{A}_{x_i,z} = [\mathbf{A} \| \mathbf{A}_{x_i} \| \mathbf{T}_1^{z[1]} \| \cdots \| \mathbf{T}_{d_z}^{z[d_z]}]$ where the last d_z matrices corresponding to the bits of d_z are public. Therefore, the group signing key of user i at time t is $\{\mathbf{S}_{i||z}, z \in \text{Nodes}_{(t \leftarrow T-1)}\}$ which satisfies $\mathbf{A}_{x_i,z} \cdot \mathbf{S}_{i||z} = \mathbf{0} \mod q$. The user is then able to compute all possible $\mathbf{S}_{i||t}$ by employing $\mathbf{S}_{i||z}$ if z is an ancestor of t where t is the binary representation of a period of time. The basis delegation technique allows users to compute trapdoor matrices for all the descendent nodes in the set $\text{Nodes}_{(t \leftarrow T-1)}$ and therefore to compute all the subsequent signing keys.

We now have introduced all the building blocks that we use to construct our group signature scheme.

4.2 Our scheme

In this section we describe our main contribution: a forward secure group signature from lattices without NIZK secure in the standard model having public key and signature size independent of the number of users for which we managed to prove forward-secure traceability and CCA-selfless anonymity. At a high level, we start from the constrained signature of Tsabary, we transform it into an ABS (according to [KY19] suggestion) as previously explained, we equip it with forward-security (following the mechanism of [Lin+19]), then plug it into the group signature of [KY19]. The drawback is that the security assumption on which the GS scheme relies is SIS with subexponential hardness.

Our main building block is then a forward secure Attribute Based Signature which is built using the idea from [Lin+19] having as starting point Tsabary's constrained signature. As explained in [Lin+19], the advantage of this method is that it incurs only logarithmic dependency on T. Therefore our construction achieves signature size and public key size constant in N and logarithmic in T. We note that [Lin+19] applied it directly for building forward-secure group signature (FS-GS) while we need to apply it first on our ABS to get forward-secure attribute-based signatures (FS-ABS). Indeed, in an encrypt-then-prove paradigm for group signatures, the transformation of [Lin+19] into a forward secure group signature is independent of the encryption scheme and of the NIZK scheme used to prove the membership. This is because the group secret key of a user does not appear as input in the NIZK proof but is embedded in a ciphertext on which the proof

is performed. Instead, the paradigm on which we build our construction uses an ABS to prove that the user belongs to the group, and the ABS secret key is a direct component of the group secret key of a user. This means that if we want to update the group secret key of a user, we need to update the ABS secret key as well.

From this observation and the fact that the ABS built by Tsabary [Tsa17] is based on lattice trapdoors which fit perfectly with bonsai trees, we can then adapt the forward security mechanism of [Lin+19] to the ABS derived from [Tsa17], and use the resulting ABS to get forward-secure group signature scheme. We note that if we try to apply the same technique for the second ABS from [KY19] (also built from lattice trapdoors) we can not get forward-security. The problem is that the design of ABS forces us to keep the initial secret key derived by the master authority for every user in order to be able to compute all the other subsequent keys for the following periods of time. This means that an adversary who gains access to a secret key for a certain period of time, would be able to compute the secret keys for all periods of time (including previous ones).

The main difficulty encountered when trying to add forward security to the ABS derived from [Tsa17] is then the way to deal with the trapdoors for each of the time periods. This includes the trapdoors considered in the ABS construction as well as in the simulation. Moreover this modification induces a new time parameter T, that has to be handled in the unforgeability proof. Indeed, the construction of [Tsa17] has been designed to only consider a fixed matrix $\mathbf{A}$ and a vector of matrices $\overrightarrow{\mathbf{A}}$ linked to the attribute to generate and verify signatures. But now we add $\log T$ additional matrices in order to integrate the time parameter, in a similar way to [Lin+19]. This transformation implies that the secret keys have to be modified according to the time period considered. It means that a trapdoor update mechanism needs to be built from the trapdoor construction of Tsabary, using tools introduced in the bonsai tree mechanism [Cas+10], and the time component has to be dealt with in the different queries from the simulation-based proof.

Finally, as we apply forward-secure property to an attribute-based construction in our case, we also have to handle an additional component which is the attribute. A naive adaptation from the transformation of [Lin+19] (on a group signature) to our construction (an attribute based signature) would not be secure. Indeed, we have to deal with two types of trapdoor: the trapdoor inherent to the ABS construction derived from [Tsa17], and the trapdoors given by the matrices linked to the time parameter. In the security proof of the ABS scheme, we need to simulate these two types of trapdoors according to each other, and according to the time period considered, in order to be able to answer all the queries

of an attacker. At the same time, we expect all these trapdoors to vanish when the forgery of the attacker is outputted, in order to be able to conclude the simulation and then to argue about the security reduction getting a solution to a hard problem.

4.2.1 A Forward Secure Attribute-based Signature on lattices

As already explained above, we equip the ABS scheme in the general construction of [KY19], which is the main component of their framework, with a forward-secure indexed ABS. We start by giving the definition and the security requirements of a forward-secure indexed attribute based signature. We note that the ABS scheme supports multiple users since it is designed as a building block for group signature scheme.

The starting point of our scheme is the constrained signature of [Tsa17]. We first adapt it into an indexed attribute-based signature, by including an index i into the attribute x, following the idea of [KY19]. Moreover we extend this construction to a forward-secure attribute-based signature scheme, by applying a transformation similar to [Lin+19]. The idea of this transformation is that we consider a pair of matrices $\mathbf{T}_j^b, b \in \{0,1\}$ for every bit j of the time period t considered. Then by concatenating these matrices $\mathbf{T}_j^b$ to the public key of [Tsa17], we can include a time period t into the verification key and the signatures. The technical difficulty that arises when using this transformation into the Tsabary's construction is simulating the secret keys for each period of time and for each user, without possessing the master secret key. This can be done by using "dummy" secret keys which vanish when the signature is made for an identity and a time period chosen selectively by the adversary at the beginning of the game, allowing the simulator to solve a hard problem (which is the SIS problem). We then get a new forward-secure attribute-based signature scheme which is independent of the number of users N, and only logarithmic on the total number of periods T.

Framework and security properties. We denote $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ the set of circuits with domain $\{0,1\}^{k(\lambda)}$ and range $\{0,1\}$. We bound the size of every circuit in $\{\mathcal{C}_\lambda\}$ by $k_c = \text{poly}(\lambda)$. We also denote the space of messages as $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, for which we bound the size elements by $k_m = \text{poly}(\lambda)$. Usually we simplify notation and just denote these spaces $\mathcal{C}$ and $\mathcal{M}$. We then define the forward-secure indexed attribute-based signature scheme for the circuit class $\mathcal{C}$:

Definition 14. *A forward-secure indexed attribute-based signature (FSI-ABS) scheme consists of the following algorithms:*

$\mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$ *The setup algorithm takes as input λ the security parameter, N the size of the index space and T the number of time periods, given in unary form, and it outputs a master public key* mpk *and a master secret key* msk.

$\mathsf{ABS.KeyGen}(\mathsf{msk}, i, x_i)$ *The key generation algorithm takes as input the master secret key* msk*, an index $i \in [N]$ and the attribute $x_i \in \{0,1\}^k$. It outputs $\mathsf{sk}_{x_i,0}$, the initial secret key associated to x_i.*

$\mathsf{ABS.KeyUpdate}(\mathsf{mpk}, i, \mathsf{sk}_{x_i,t}, t+1)$ *The key update algorithm takes as input the master secret key* msk*, an index of an user i as well as its secret key for the time t,* $\mathsf{sk}_{x_i,t}$*. It updates this key $\mathsf{sk}_{x_i,t}$ for the next time period $t+1$ and outputs $\mathsf{sk}_{x_i,t+1}$.*

$\mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C, M, t)$ *The signing algorithm takes as input the master public key* mpk*, a secret key $\mathsf{sk}_{x_i,t}$ for the current period of time t, a circuit $C \in \mathcal{C}_\lambda$, a message $M \in \mathcal{M}_\lambda$ and a time period t and it outputs an attribute-based signature σ if $C(x_i) = 0$.*

$\mathsf{ABS.Verify}(\mathsf{mpk}, C, M, \sigma, t)$ *The verification algorithm takes as input the master public key* mpk*, the circuit $\mathcal{C}$, the message M, the attribute-based signature σ and the time period t. This algorithm outputs* Valid *if the signature σ is valid for the time period t and* $\mathsf{Invalid}$ *otherwise.*

For a FSI-ABS scheme, we require *correctness* and two security properties: *perfect-privacy* and *forward-secure policy-selective unforgeability*. Perfect privacy captures the idea that the attribute used to sign a message must remain anonymous. The unforgeability property says that even if users collude they can not forge a signature on a message associated with a policy if none of the attributes satisfies the policy. We next develop the formal definitions of these three properties. We note that we can not achieve selective unforgeability directly, but we start from no-signing-query and apply a transformation usign admissible hash functions to obtain selective unforgeability.

Correctness. We require that for all $\lambda, N \in \mathsf{poly}(\lambda), T \in \mathbb{N}, t \in [T]$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$, $i \in [N]$, $x_i \in \{0,1\}^k, C \in \mathcal{C}_\lambda$ such that $C(x_i) = 0$, $M \in \mathcal{M}_\lambda, \mathsf{sk}_{x_i,0} \leftarrow \mathsf{ABS.KeyGen}(\mathsf{msk}, i, x_i)$, $\mathsf{sk}_{x_i,t} \leftarrow \mathsf{ABS.KeyUpdate}(\mathsf{mpk}, i, \mathsf{sk}_{x_i,t-1}, t)$, $\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C, M, t)$, we have that $\mathsf{ABS.Verify}(\mathsf{mpk}, C, M, \sigma, t) = \mathsf{Valid}$.

Perfect privacy. A FSI-ABS scheme provides **perfect privacy** if for all $\lambda, N \in \text{poly}(\lambda)$, $T \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$, $x_0, x_1 \in \{0,1\}^k$, $i_0, i_1 \in [N]$, $C \in \mathcal{C}_\lambda$, $t \in [T]$, $C(x_0) = C(x_1) = 0$, $M \in \mathcal{M}_\lambda$,
$\mathsf{sk}_{x_b,0} \leftarrow \mathsf{ABS.KeyGen}(\mathsf{msk}, i_b, x_b)$ and $\mathsf{sk}_{x_b,t} \leftarrow \mathsf{ABS.KeyUpdate}(\mathsf{mpk}, b, \mathsf{sk}_{x_b,t-1}, t)$, the distributions $\mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_0,t}, C, M, t)$ and $\mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_1,t}, C, M, t)$ are indistinguishable.

Forward-secure policy-selective unforgeability. We define the unforgeability property for an indexed attribute-based signature scheme following the framework from [Yue+12]. We use a game model between a challenger and an attacker $\mathcal{A}$ to define this security notion.

Setup: At the beginning of the game, the adversary $\mathcal{A}$ is given $1^\lambda, 1^N, 1^T$ as input. It then sends to the challenger the tuple (C^*, M^*, t^*) consisting of a circuit, a message and a time period for which he is going to forge a signature. The challenger gets $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$. It gives mpk to $\mathcal{A}$. At the start of each time period $t \in [T]$, the challenger announces the beginning of t to $\mathcal{A}$. During current time period t, the challenger responds to $\mathcal{A}$'s queries as follows:

Key Queries: $\mathcal{A}$ sends (i, x_i, t) to the challenger and gets back $\mathsf{sk}_{x_i,t}$.

Signing Queries: $\mathcal{A}$ can perform some signing queries to the challenger during the game. If $\mathcal{A}$ queries (C, M, t, i), with $M \in \mathcal{M}, C \in \mathcal{C}, i \in [N]$ and $C(x_i) = 0$, the challenger generates $\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C, M, t)$ and sends it to $\mathcal{A}$.

Forgery: Eventually $\mathcal{A}$ outputs $(C^*, M^*, \sigma^*, t^*)$ as a forgery. Then $\mathcal{A}$ wins the game if:

1. $C^* \in \mathcal{C}$,
2. $(C^*, M^*, t^*, \cdot)$ was not queried in a Signing query,
3. $\mathsf{ABS.Verify}(\mathsf{mpk}, C^*, M^*, \sigma^*, t^*) = \mathsf{Valid}$,
4. $C^*(x_i) = 1$ for any key queried by $\mathcal{A}$ respective to t and x_i where $t \leq t^*$.

We then define the advantage of the attacker $\mathcal{A}$ against forward-secure policy-selective unforgeability as the probability that $\mathcal{A}$ wins the above game. We say that a scheme satisfies the forward-secure policy-selective unforgeability property, if for all PPT adversary, the advantage in the above game is negligible.

In our ABS construction, we additionally make use of two different flavours of unforgeability as in [KY19]:

— **No-signing-query unforgeability:** The game model of the no-signing-query unforgeability is the same as policy-selective unforgeability, but the attacker can not perform signature queries.

— **Adaptive unforgeability:** The game model of the adaptive unforgeability is the same as the policy-selective unforgeability, but the attacker is not asked anymore to give the tuple (C^*, M^*, t^*) at the beginning of the game, he can rather chooses it adaptively during the game.

Construction of FSI-ABS scheme from lattices. We adapt the constrained signature developed by Tsabary [Tsa17] to a forward-secure attribute-based signature scheme. As explained by Katsumata and Yamada [KY19], the signature scheme of Tsabary is not an attribute-based signature but a constrained signature. It means that in the constrained signature, a user does not sign a circuit but an attribute. Then the role of the attribute and the circuit are exchanged compared to an actual attribute-based signature scheme. However, as explained in [KY19], we can turn a constrained signature into an attribute-based signature: we consider a constraint space composed of all d-depth bounded circuit $\mathcal{F}_d = \{f{:}\{0,1\}^\ell \to \{0,1\}\}$, with $\ell = \text{poly}(\lambda)$, then a constraint f can be seen as a universal circuit $U(\cdot,\cdot,x)$ (that we denote $U_x(\cdot,\cdot)$), which takes as input the circuit-message pair (C, M) (seen as a string of size ℓ).

Our contribution is to build a forward-secure attribute-based signature scheme meaning that the lifetime of the scheme is divided into $T = 2^d$ discrete periods. To represent the time periods we use a binary tree, then each time period t is associated with a leaf $\mathsf{Bin}(t)$. Following [Boy+06], for $j \in [d+1]$, we define a time period's "second sibling at depth j". Intuitively, it corresponds to the right neighbour at depth j of each node on the path from the root of the tree to the leaf $\mathsf{Bin}(t)$.

$$\mathsf{Sibling}(j,t) = \left\{ \begin{array}{ll} (1) & \text{if } j = 1 \text{ and } \mathsf{Bin}(t)[j] = 0 \\ (\mathsf{Bin}(t)[1], ..., \mathsf{Bin}(t)[j-1], 1) & \text{if } 1 < j \leq d \text{ and } \mathsf{Bin}(t)[j] = 0 \\ \perp & \text{if } 1 \leqslant j \leq d \text{ and } \mathsf{Bin}(t)[j] = 1 \\ \mathsf{Bin}(t) & \text{if } j = d+1 \end{array} \right\}.$$

We also define node set $\text{Nodes}_{(t \to T-1)}$ to be $\{\mathsf{Sibling}(1,t), ..., \mathsf{Sibling}(d+1,t)\}$. The goal of this set is to uniquely define the path to each leaf of the tree.

We consider also a function called bitstr which takes as input a message-circuit pair (C, M) and which outputs its input seen as a string of bits. Then:

bitstr: $\{0,1\}^{k_c} \times \{0,1\}^{k_m} \mapsto \{0,1\}^{\ell}$, such that $\mathsf{bitstr}(C, M) = \{C_1, \cdots, C_{k_c}, M_1, \cdots, M_{k_m}.\}$

ABS.Setup$(1^\lambda, 1^N, 1^T)$ On input the security parameter 1^λ, 1^N where N is the number of indexes $i \in [N]$ and 1^T where T is the number of time periods $T = 2^d$ for some $d \in \mathbb{N}$, it sets the parameters n, m, p, q, γ_0 to be polynomial in λ. Then, it generates:

- uniform matrices $\overrightarrow{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n \times \ell m}$,
- $(\mathbf{A}, \mathbf{A}_{\gamma_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^p, q)$, with $\mathbf{A} \in \mathbb{Z}_q^{n \times p}$ and $\mathbf{A}_{\gamma_0}^{-1}$ its trapdoor,
- $2d$ matrices $\mathbf{T}_j^b \xleftarrow{\$} \mathbb{Z}_q^{n \times p}$ for all $j \in [d]$ and $b \in \{0,1\}$.

It algorithm outputs: $\mathsf{mpk} = (\mathbf{A}, \overrightarrow{\mathbf{A}}, \{\mathbf{T}_j^b\}_{j \in [d], b \in \{0,1\}})$ and $\mathsf{msk} = (\mathbf{A}_{\gamma_0}^{-1})$.

ABS.KeyGen(msk, i, x_i) On input the master secret key msk, the index $i \in [N]$ and the attribute $x_i \in \{0,1\}^k$, it computes U_{x_i}, $\mathbf{H}_{x_i} = \mathsf{EvalF}(U_{x_i}, \overrightarrow{\mathbf{A}}) \in \mathbb{Z}_q^{\ell m \times m}$ as defined in Theorem 2 and $\mathbf{A}_{x_i} = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_{x_i} \in \mathbb{Z}_q^{n \times m}$. Then, it uses $\mathbf{A}_{\gamma_0}^{-1}$ to compute $\mathbf{R}_{x_i} = [\mathbf{A} \| \mathbf{A}_{x_i}]^{-1}$. Then it determines the set $\mathrm{Nodes}_{(0 \to T-1)}$ and for $z \in \mathrm{Nodes}_{(0 \to T-1)}$:

- if $z = \perp$, set $\mathsf{sk}_{x_i}[z] = \perp$,
- else it denotes d_z as the bit-length of z, with $d_z \leqslant d$, and computes the matrix: $\mathbf{A}_{x_i,z} = [\mathbf{A} \| \mathbf{A}_{x_i} \| \mathbf{T}_1^{\mathsf{Bin}(z)[1]} \| \cdots \| \mathbf{T}_{d_z}^{\mathsf{Bin}(z)[d_z]}] \in \mathbb{Z}_q^{n \times ((d_z+1)p+m)}$, then it computes: $\mathbf{R}_{x_i,z} \leftarrow \mathrm{RandBasis}(\mathrm{ExtBasis}(\mathbf{R}_{x_i}, \mathbf{A}_{x_i,z}), s_{d_z})$, and set $\mathsf{sk}_{x_i}[z] = \mathbf{R}_{x_i,z}$,

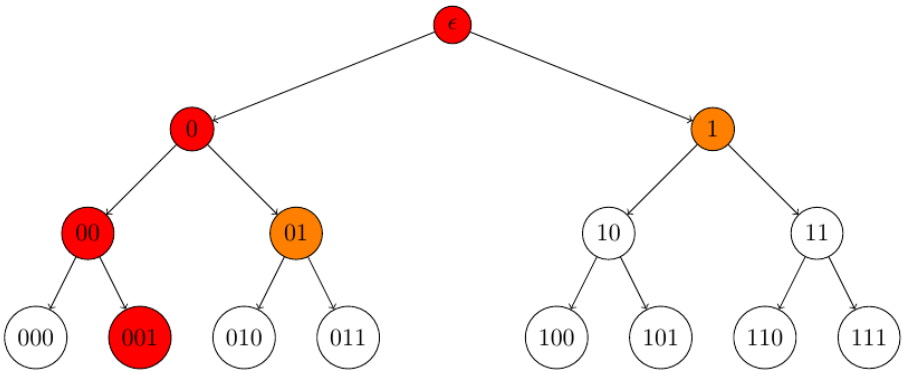

Figure 4.1 – A binary tree with time periods $\mathrm{T} = 2^3$. In order to fill the set $\mathrm{Nodes}_{(t \to T-1)}$ we begin with the leaf $\mathrm{Bin}(t)$ that we add in the set $\mathrm{Nodes}_{(t \to T-1)}$, together with its sibling (which is its right neighbour), if it exists. Then recursively, we go up in the tree to the parent of the node considered (coloured in red), and we add its sibling (coloured in orange) to the set $\mathrm{Nodes}_{(t \to T-1)}$ (still if it exists). We keep going this way, until we reach the root of the binary tree. We stop then and output the corresponding list $\mathrm{Nodes}_{(t \to T-1)}$. On the path from node ϵ to the leaf node (001) we then have $\mathrm{Nodes}_{(1 \to 7)} = \{(1), (01), \perp, (001)\}$.

Finally we get: $\mathsf{sk}_{x_i,0} = \{\mathsf{sk}_{x_i}[z], z \in \text{Nodes}_{(0\to T-1)}\}$.

ABS.KeyUpdate$(\mathsf{mpk}, i, \mathsf{sk}_{x_i,t}, t+1)$ First parse the set $\mathsf{sk}_{x_i,t} = \{\mathsf{sk}_{x_i}[z], z \in \text{Nodes}_{(t\to T-1)}\}$ and determine the set $\text{Nodes}_{(t+1\to T-1)}$.

For $z' \in \text{Nodes}_{(t+1\to T-1)}$:

- if $z' = \perp$, set $\mathsf{sk}_{x_i}[z'] = \perp$.
- Otherwise, there exists exactly one $z \in \text{Nodes}_{(t\to T-1)}$ which is a prefix of z' i.e. $z' = z\|y$. There are two possibilities here:
  1. if $z' = z$ then $\mathsf{sk}_{x_i}[z'] = \mathsf{sk}_{x_i}[z]$,
  2. if $z' = z\|y$ for some non-empty y, then z is an ancestor of z', and from $\mathsf{sk}_{x_i}[z] = \mathbf{R}_{x_i,z}$ it can delegate a basis
     $\mathbf{R}_{x_i,z'} \leftarrow \text{RandBasis}(\text{ExtBasis}(\mathbf{R}_{x_i,z}, \mathbf{A}_{x_i,z'}), s_{d_{z'}})$, and set
     $\mathsf{sk}_{x_i}[z'] = \mathbf{R}_{x_i,z'}$.

Finally output $\mathsf{sk}_{x_i,t+1} = \{\mathsf{sk}_{x_i}[z'], z' \in \text{Nodes}_{(t+1\to T-1)}\}$.

ABS.Sign$(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C, M, t)$ First compute $\mathbf{x} = \mathsf{bitstr}(C, M)$. If $U_{x_i}(\mathbf{x}) = C(x_i) \neq 0$ output $\perp$. Otherwise, first compute $\mathbf{H}_{x_i,\mathbf{x}} = \mathsf{EvalFX}(U_{x_i}, \mathbf{x}, \overrightarrow{\mathbf{A}}) \in \mathbb{Z}_q^{\ell m \times m}$, as defined in Theorem 2, such that $(\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{x_i,\mathbf{x}} = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_{x_i} - U_{x_i}(\mathbf{x})\mathbf{G} = \mathbf{A}_{x_i}$ as $U_{x_i}(\mathbf{x}) = 0$.
Compute $\overrightarrow{\mathbf{B}_t} = [\mathbf{A}\|\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}\|\mathbf{T}_1^{\mathsf{Bin}(t)[1]}\|\cdots\|\mathbf{T}_d^{\mathsf{Bin}(t)[d]}] \in \mathbb{Z}_q^{n\times((d+1)p+\ell m)}$, and

$$\mathbf{S}_i = \begin{bmatrix} \mathbf{I}_p & & & & \\ & \mathbf{H}_{x_i,\mathbf{x}} & & & \\ & & \mathbf{I}_p & & \\ & & & \cdots & \\ & & & & \mathbf{I}_p \end{bmatrix} \in \mathbb{Z}_q^{((d+1)p+\ell m)\times((d+1)p+m)}.$$

We then have $\overrightarrow{\mathbf{B}_t} \cdot \mathbf{S}_i = [\mathbf{A}\|\mathbf{A}_{x_i}\|\mathbf{T}_1^{\mathsf{Bin}(t)[1]}\|\cdots\|\mathbf{T}_d^{\mathsf{Bin}(t)[d]}] = \mathbf{A}_{x_i,t}$. Since $\mathsf{sk}_{x_i,t}$ contains a trapdoor for $\mathbf{A}_{x_i,t}$, we can apply the trapdoor extension from Lemma 5 to obtain $\mathbf{B}_{\tau_u}^{-1} = [\overrightarrow{\mathbf{B}_t}]^{-1} = [\mathbf{A}\|\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}\|\mathbf{T}_1^{\mathsf{Bin}(t)[1]}\|\cdots\|\mathbf{T}_d^{\mathsf{Bin}(t)[d]}]_{\tau_u}^{-1}$, where $\mathbf{A} = \mathbf{A}_{x_i,t}$, $\mathbf{B} = \overrightarrow{\mathbf{B}_t}$ and $\mathbf{S} = \mathbf{S}_i$ using $\mathsf{sk}_{x_i,t} = [\mathbf{A}_{x_i,t}]_{\tau_s}^{-1}$.

Then the signer has a trapdoor for $\overrightarrow{\mathbf{B}_t}$ and he can compute $\sigma_{\mathbf{x},t} \xleftarrow{\$} \overrightarrow{\mathbf{B}_t}^{-1}(\mathbf{0})$.

ABS.Verify$(\mathsf{mpk}, C, M, \sigma_{\mathbf{x},t}, t)$. First, compute $\mathbf{x} = \mathsf{bitstr}(C, M)$ and then check that:

- $[\mathbf{A}\|\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}\|\mathbf{T}_1^{\mathsf{Bin}(t)[1]}\|\cdots\|\mathbf{T}_d^{\mathsf{Bin}(t)[d]}] \cdot \sigma_{\mathbf{x},t} = \mathbf{0}$,

- $\|\sigma_{\mathbf{x},t}\|_\infty \leqslant B$.

If the verification passes, then output Valid, if not, output Invalid.

Selection of parameters for our FSI-ABS scheme. We end this part by giving a selection of parameters that fit with the above construction and with the security requirement, that we expect to be fulfill.

Given the security parameter λ, the parameters m_0, p, γ_0 and τ_s are chosen according to TrapGen algorithm, $T = 2^d$ is chosen as a power of 2, for $d \in \mathbb{N}$, and is the number of time periods considered, and ℓ is the size of input of the circuit. We choose parameters τ_u and B by referring to Theorem 2. Finally s_j is dictated also by Lemma 5. Then we set:

— $m = 4n\lceil \log q \rceil$,
— $m_0 = O(n \log q) \geqslant 4n \log q$,
— $p = \max\{m_0, (n+1)\lceil \log q \rceil + 2\lambda\}$,
— $\gamma_0 = O(\sqrt{n\lceil \log q \rceil \log n})$,
— $\tau_s = \max\{\sqrt{p} \cdot \ell \cdot 2^d m^{1.5+d}, \gamma_0\}$,
— $\tau_u = \tau_s \cdot \sqrt{\ell} \cdot 2^d m^{0.5+d}$,
— $B = \tau_u \sqrt{(1+d) \cdot p + \ell \cdot m}$,
— $s_j = \mathcal{O}(\sqrt{nd \log q})^{j+1} \cdot \omega(\sqrt{\log n})^{j+1}$ for $j \in [d]$.

4.2.2 Our group signature scheme construction

In this section we present the construction of our forward-secure group signature (FS-GS) scheme from lattices. We use the model of forward-secure group signature scheme formalized in [NHF09] and [Lin+19]. We give the formal definition of FS-GS, correctness and security properties - *CCA-selfess anonymity* and *forward-secure traceability* - below.

Definition 15. *A forward-secure group signature scheme consists of the following algorithms:*

GS.KeyGen$(1^\lambda, 1^N, 1^T)$ *is a randomized algorithm taking as input a security parameter λ, number of users N and number of time periods T. Its output consists of a group public key* gpk*, an opening key* gok *and a set of initial user secret keys* $\{\mathsf{gsk}_{i,0}\}_{i\in[N]}$.

GS.KeyUpdate$(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, t+1)$ *is a randomized algorithm that takes as input the group public key* gpk*, the secret key* $\mathsf{gsk}_{i,t}$ *of user i at time t, a user i and a time period $t+1$ and outputs* $\mathsf{gsk}_{i,t+1}$*, the secret signing key of user i at time $t+1$.*

$\mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t)$ *takes as input the group public key* gpk*, the* i*th user secret key* $\mathsf{gsk}_{i,t}$ *at time* t*, the index* i *of the user, a message* $M \in \{0,1\}^*$ *and the current time interval* t *and outputs a group signature* Σ*.*

$\mathsf{GS.Verify}(\mathsf{gpk}, M, \Sigma, t)$ *takes as input the group public key* gpk*, a message* M*, a signature* Σ *and the time period* t*. It outputs either* Valid *or* $\mathsf{Invalid}$*.* Valid *indicates that* Σ *is a valid signature on* M *at time period* t *w.r.t* gpk*.*

$\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M, \Sigma, t)$ *takes as input the group public key* gpk*, the opening key* gok*, a message* M*, a signature* Σ *and time interval* t *and outputs an identity or* $\mathsf{Invalid}$ *if it fails to identify the signer.*

We require two security properties: *forward-secure traceability* and *CCA-selfless anonymity* besides the *correctness* property.

Correctness. We require that for all $\lambda, N \in \text{poly}(\lambda), T \in \mathbb{N}$, $(\mathsf{gpk}, \mathsf{gok}, \{\mathsf{gsk}_{i,0}\}_{i\in[N]}) \leftarrow \mathsf{GS.KeyGen}(1^\lambda, 1^N, 1^T)$, $\forall i \in [N]$, all $M \in \{0,1\}^*$, all $\mathsf{gsk}_{i,t} \leftarrow \mathsf{GS.KeyUpdate}(\mathsf{gpk}, \mathsf{gsk}_{i,t-1}, i, t)$ and for all $t \in [T]$, the following equations hold:

$$\mathsf{GS.Verify}(\mathsf{gpk}, M, \mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t), t) = \mathsf{Valid}, \text{ and}$$
$$\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M, \mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t), t) = i.$$

CCA-selfless anonymity. We say that a forward-secure group signature scheme provides **CCA-selfless anonymity** if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the following game.

Setup: At the begining of the game, adversary $\mathcal{A}$ is given $1^\lambda, 1^N, 1^T$ as input and sends $i_0^*, i_1^* \in [N]$ to the challenger. The challenger runs $\mathsf{GS.KeyGen}(1^\lambda, 1^N, 1^T)$ to produce a public key gpk, a secret key gok and users secret keys $\mathsf{gsk} = \{\mathsf{gsk}_{i,0}\}_{i\in[N]}$ and gives $(\mathsf{gpk}, \{\mathsf{gsk}_{i,0}\}_{i\in[N]\setminus\{i_0^*, i_1^*\}})$ to $\mathcal{A}$.

Queries: At the begining of each time period, the challenger increments a counter t and notifies $\mathcal{A}$ about it. During current time interval t, $\mathcal{A}$ can make the following queries unbounded polynomially many times.

Signing: On input index $b \in \{0,1\}$ and message M, the challenger generates and outputs a signature Σ generated for member i_b and period t as $\Sigma \leftarrow \mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i_b^*,t}, i, M, t)$

Opening: When receiving a query (M, Σ, t) from $\mathcal{A}$, the challenger runs $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M, \Sigma, t)$ and returns the result to $\mathcal{A}$.

Challenge phase: At some period $t^* \in \{1, .., T\}$, $\mathcal{A}$ chooses its target message M^*. The challenger then flips a coin $d^* \xleftarrow{\$} \{0, 1\}$, computes and returns $\mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i^*_{d^*}, t^*}, i^*_{d^*}, M^*, t^*)$ to $\mathcal{A}$.

Queries: After the challenge phase, $\mathcal{A}$ may continue to make signing and opening queries unbounded polynomially many times. She may not make an open query for (M^*, Σ^*, t^*).

Guess: Eventually, $\mathcal{A}$ outputs d' and wins if $d' = d^*$.

We define the advantage of $\mathcal{A}$ as $|Pr[d' = d^*] - 1/2|$ where the probability is taken over the randomness of the challenger and the adversary. A forward-secure group signature scheme is said to be CCA-selfless anonymous if the advantage of any adversary $\mathcal{A}$ is negligible in the above game.

Forward-secure traceability. A group signature scheme has the **forward-secure traceability** property if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the following game where he maintains a list $\mathcal{CU}$ which is set to be empty at the beginning of the game.

Setup: At the beginning of the game, the challenger runs $\mathsf{GS.KeyGen}$ and obtains $(\mathsf{gpk}, \mathsf{gok}, \{\mathsf{gsk}_{i,0}\}_{i \in [N]})$. The adversary $\mathcal{A}$ is given $(\mathsf{gpk}, \mathsf{gok})$.

Queries: During the game, $\mathcal{A}$ can make the following queries unbounded polynomially many times.

Signing: On input index $i \in \{1, .., N\}$, a message M and a period time t, the challenger generates and outputs a signature Σ generated for member i and period t as $\Sigma \leftarrow \mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t)$ if $(i, t) \notin \mathcal{CU}$.

Corrupt: Given an index i and time moment t, the challenger returns $\mathsf{gsk}_{i,t}$ to $\mathcal{A}$ if $(i, t) \notin \mathcal{CU}$. The challenger adds (i, t) to $\mathcal{CU}$.

Forgery: $\mathcal{A}$ eventually comes up with a signature Σ^* on a message M^* and a time period t^*. We say that $\mathcal{A}$ wins the game if:

1. $\mathsf{GS.Verify}(\mathsf{gpk}, M^*, \Sigma^*, t^*) \rightarrow \mathsf{Valid}$.
2. Only one of the following two conditions is satisfied concerning the execution of the opening algorithm where $i^* = \mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*)$:

(a) The opening algorithm fails i.e. $i^* = \mathsf{Invalid}$

(b) The message M^* was not sent in a signing query before and one of the following is true: $(i^*, t^*) \notin \mathcal{CU}$ or $(i^*, t^*) \in \mathcal{CU}$ but $\mathcal{A}$ did not obtain $\mathsf{gsk}_{i^*,t}$ such that $t \leq t^*$.

We define the advantage of $\mathcal{A}$ as the probability that he wins the above game, where the probability is taken over the randomness of the challenger and the adversary. A GS scheme is said to satisfy forward-secure traceability if the advantage of any adversary $\mathcal{A}$ is negligible in the above game.

Forward-secure group signature from lattices. We now describe our lattice-based FS-GS scheme which employs the FSI-ABS scheme given in the previous section and which satisfies CCA-selfless anonymity and traceability. As the ABS used is forward-secure, we show that the group signature is also forward-secure, so we consider that the lifetime of the scheme is divided into T time periods. When entering a new period of time, a new secret key is computed from the current one and afterwards the current key is deleted promptly.

GS.KeyGen$(1^\lambda, 1^N, 1^T)$ On input security parameter λ, the number of group members N and the total number of time periods $T = 2^d$, the algorithms works as follows:

— sample $\mathsf{pp} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$,

— for $i \in [N]$, sample $K_i \leftarrow \mathsf{SKE.Gen}(\mathsf{pp})$ and compute $sk_{x_i,0}$ as $sk_{i||K_i,0} \leftarrow \mathsf{ABS.KeyGen}(\mathsf{msk}, i, i||K_i)_{i\in[N]}$,

— output $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}), \mathsf{gok} = \{K_i\}_{i\in[N]}, \mathsf{gsk}_{i,0} = (i, K_i, \mathsf{sk}_{i||K_i,0})$.

GS.KeyUpdate$(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, t+1)$ It calls the key update algorithm of the ABS and returns $\mathsf{gsk}_{i,t+1} = (i, K_i, \mathsf{ABS.KeyUpdate}(\mathsf{mpk}, i, \mathsf{sk}_{t,i}, t+1))$.

GS.Sign$(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t)$ In order to sign a message, the user samples $(\mathsf{ovk}, \mathsf{osk}) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$ and computes the encryption of his identity under the key K_i as $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(K_i, i||\mathsf{ovk})$. Then, he computes

$$\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{i||K_i}, C[\mathsf{ovk}, \mathsf{ct}], M, t),$$

where the circuit $C[\mathsf{ovk}, \mathsf{ct}]$ is defined as follows:

$C[\mathsf{ovk}, \mathsf{ct}](i||K_i)$

Hardwired constants: a verification key ovk of OTS and ciphertext ct of SKE

— Retrieve $i \in [N]$ and K_i from the input. If this is impossible, return 1.

— Compute $\mathsf{SKE.Dec}(K_i, \mathsf{ct}) = i'||\mathsf{ovk}'$. If $i' = i$ and $\mathsf{ovk}' = \mathsf{ovk}$ output 0. Otherwise, output 1.

Finally run $\tau \leftarrow \mathsf{OTS.Sign}(\mathsf{osk}, M\|\sigma)$.

The signature consists of $\Sigma = (\mathsf{ct}, \mathsf{ovk}, \sigma, \tau)$.

GS.Verify$(\mathsf{gpk}, M, \Sigma, t)$ On input gpk, a message M, a group signature Σ on M and a period time t, check that $\mathsf{ABS.Verify}(\mathsf{mpk}, C[\mathsf{ovk}, \mathsf{ct}], M, \sigma, t) = \mathsf{Valid}$ and $\mathsf{OTS.Verify}(\mathsf{ovk}, \tau, M\|\sigma) = \mathsf{Valid}$; if one of these verification condition does not hold, return $\mathsf{Invalid}$. Otherwise return Valid.

GS.Open$(\mathsf{gpk}, \mathsf{gok}, M, \Sigma, t)$ First run **GS.Verify**$(\mathsf{gpk}, M, \Sigma, t)$ and return $\mathsf{Invalid}$ if the verification result does not hold. Otherwise, parse $\Sigma \rightarrow (\mathsf{ct}, \mathsf{ovk}, \sigma, \tau)$. Since the manager does not know the identity of the user who produced the signature, he has to find it by trial and error, i.e. he computes $d_i \leftarrow \mathsf{SKE.Dec}(K_i, \mathsf{ct})$ for $i \in [N]$ and outputs the smallest index i such that $d_i \neq \mathsf{Invalid}$. If there is no such i, return $\mathsf{Invalid}$.

4.2.3 Analysis and security

Now we developed the construction of our group signature scheme as well as our attribute-based signature scheme, we prove that they are both secure. Below we begin by giving the proofs of correctness, privacy and unforgeability for the ABS scheme.

Correctness. We fix an attribute $x_i \in \{0,1\}^k$ for user $i \in [N]$, a circuit $C \in \mathcal{C}_\lambda$ such that $C(x_i) = 0$, a time period t and a message M and let $\mathbf{x} = \mathsf{bitstr}(C, M)$. Consider $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$ and
$\sigma_{\mathbf{x},t} \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{ABS.KeyUpdate}(\mathsf{mpk}, i, \mathsf{sk}_{x_i,t-1}, t), C, M, t)$, since the signature $\sigma_{\mathbf{x},t} \neq \perp$ because of $C(x_i) = 0$, we have that:
$\sigma_{\mathbf{x},t} \in \mathbf{B}_{\tau_u}^{-1}(\mathbf{0}) = [\mathbf{A}\|\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}\|\mathbf{T}_1^{\mathsf{Bin}(t)[1]}\|\cdots\|\mathbf{T}_d^{\mathsf{Bin}(t)[d]}]_{\tau_u}^{-1}(\mathbf{0})$ and
$[\mathbf{A}\|\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}\|\mathbf{T}_1^{\mathsf{Bin}(t)[1]}\|\cdots\|\mathbf{T}_d^{\mathsf{Bin}(t)[d]}] \cdot \sigma_{\mathbf{x},t} = \mathbf{0}$. Moreover, we know from lattice trapdoor properties that samples from $\mathbf{B}_{\tau_u}^{-1}(\mathbf{0})$ have discrete Gaussian distribution over $\mathbb{Z}_q^{p+\ell m+dp}$

with parameter τ_u and, therefore, $\|\sigma_{\mathbf{x},t}\|_\infty \leq \tau_u\sqrt{(1+d)\cdot p + \ell \cdot m} = B$ and $\mathsf{ABS.Verify}(\mathsf{mpk}, C, M, \sigma_{\mathbf{x},t}, t) = \mathsf{Valid}$.

Lemma 10. *Our FSI-ABS scheme is perfectly private.*

Proof. We consider the perfect privacy game defined in Section 4.2.1. We change the way each signature $\sigma_{\mathbf{x},b}$ is generated, with $\mathbf{x} = \mathsf{bitstr}(C, M), b \in \{0,1\}$. We use the trapdoor $\mathbf{A}_{\gamma_0}^{-1}$ to compute $[\mathbf{A}\|\overrightarrow{\mathbf{A}} - \mathbf{x}\otimes\mathbf{G}\|\mathbf{T}_1^{\mathsf{Bin}(t)[1]}\|\cdots\|\mathbf{T}_d^{\mathsf{Bin}(t)[d]}]_{\tau_u}^{-1}$ (we have $\tau_u > \gamma_0$). Then using this trapdoor, we compute $\sigma_{\mathbf{x},t,b} = [\mathbf{A}\|\overrightarrow{\mathbf{A}} - \mathbf{x}\otimes\mathbf{G}\|\mathbf{T}_1^{\mathsf{Bin}(t)[1]}\|\cdots\|\mathbf{T}_d^{\mathsf{Bin}(t)[d]}]_{\tau_u}^{-1}(\mathbf{0}), b \in \{0,1\}$. This modification does not change the distribution for each $\sigma_{\mathbf{x},t,b}, b \in \{0,1\}$, which means that this change is statistically indistinguishable. Now the signature generation is totally independent of the bit b. Then the ABS scheme is perfectly private.

□

Lemma 11. *Our FSI-ABS satisfies forward-secure no-signing-query unforgeability assuming* $\mathsf{SIS}_{n,q,B',m'}$ *is hard, with* $B' = (\ell(m+d)+1)B$ *and* $m' = (d+1)p + \ell\cdot m$.

Proof. To prove this lemma, we will show that for any PPT adversary $\mathcal{A}$ against the forward secure no-signing-query unforgeability of the ABS scheme with advantage ϵ, we can build a PPT algorithm $\mathcal{B}$ that solves SIS with probability at least $\epsilon - negl(\lambda)$. Therefore, by assuming the hardness of the SIS problem, we conclude that ϵ is negligible. The proof is done in a sequence of games where the first game is identical to the forward-secure no-signing-query unforgeability game.

Game 0**:** The first game is the forward-secure no-signing-query unforgeability game between adversary $\mathcal{A}$ and the challenger.

Game 1**:** In this game, we change the way the public matrices $\overrightarrow{\mathbf{A}}$ and $\mathbf{T}_j^b$ (with $b \in \{0,1\}$ and $j \in [d]$) are generated. Upon receiving (M^*, C^*, t^*), the challenger denotes $\mathbf{y}^* = \mathsf{bitstr}(C^*, M^*)$ and does the following:

— Generates $(\mathbf{A}, \mathbf{A}_{\gamma_0}^{-1})$ as before and then samples $\overrightarrow{\mathbf{R}}_A \xleftarrow{\$} \{0,1\}^{p\ell\times m}$ and computes $\overrightarrow{\mathbf{A}} = \mathbf{A}\overrightarrow{\mathbf{R}}_A + \mathbf{y}^*\otimes\mathbf{G}$, the new distribution of $\overrightarrow{\mathbf{A}}$ is at negligible distance from the uniform distribution on $\mathbb{Z}^{n\times\ell m}$ thanks to Lemma 3 (as $p \geq 4n\log q$),

— Generates $\mathbf{R}_j^{\mathsf{Bin}(t^*)[j]} \xleftarrow{\$} \{0,1\}^{p\times p}$ and computes $\mathbf{T}_j^{\mathsf{Bin}(t^*)[j]} = \mathbf{A}\mathbf{R}_j^{\mathsf{Bin}(t^*)[j]}$, using Lemma 3, the actual distribution of $\mathbf{T}_j^{\mathsf{Bin}(t^*)[j]}$ is at negligible distance from the uniform distribution on $\mathbb{Z}_q^{n\times p}$,

— Generates $\mathbf{T}_j^{1-\mathsf{Bin}(t^*)[j]}$ for all $j \in [d]$ via $(\mathbf{T}_j^{1-\mathsf{Bin}(t^*)[j]}, \mathbf{S}_j) \leftarrow \mathsf{TrapGen}(1^n, 1^p, q)$, the new distribution of $\mathbf{T}_j^{1-\mathsf{Bin}(t^*)[j]}$ is at negligible distance from the uniform distribution thanks to Lemma 4.

Finally, we can argue that the Game 0 and Game 1 are statistically indistinguishable.

Game 2**:** In this game, we change the way the challenger answers key queries. For a key query (i, x_i, t), there are two possibilities:

$t \leq t^*$**:** In this case, from the conditions that $\mathcal{A}$ has to meet in order to win the forward-secure no-signing-query selective unforgeability game after outputting the forgery, the attacker can not query a key on an attribute x_i such that $U_{x_i}(\mathbf{y}^*) = C^*(x_i) = 0$, then $U_{x_i}(\mathbf{y}^*) = 1$.

By Theorem 2, we have $(\overrightarrow{\mathbf{A}} - \mathbf{y}^* \otimes \mathbf{G}) \cdot \mathbf{H}_{x_i,\mathbf{y}^*} = \mathbf{A}_{x_i} - U_{x_i}(\mathbf{y}^*)\mathbf{G} = \mathbf{A}_{x_i} - \mathbf{G}$. Then, we have $\mathbf{A}_{x_i} = (\overrightarrow{\mathbf{A}} - \mathbf{y}^* \otimes \mathbf{G}) \cdot \mathbf{H}_{x_i,\mathbf{y}^*} + \mathbf{G} = \mathbf{A} \cdot \overrightarrow{\mathbf{R}}_A \cdot \mathbf{H}_{x_i,\mathbf{y}^*} + \mathbf{G}$. This allows (by Lemma 8) to compute $\mathbf{R}_{x_i} = [\mathbf{A}||\mathbf{A}_{x_i}]_{\tau_s}^{-1} = [\mathbf{A}||\mathbf{A} \cdot \overrightarrow{\mathbf{R}}_A \cdot \mathbf{H}_{x_i,\mathbf{y}^*} + \mathbf{G}]_{\tau_s}^{-1}$ given $\mathbf{A}, \overrightarrow{\mathbf{R}}_A$ and $\mathbf{H}_{x_i,\mathbf{y}^*}$. We remark that the parameters from Lemma 8 are satisfied as:

$$\begin{cases} \|\mathbf{H}_{x_i,\mathbf{y}^*}\|_\infty \leq (2m)^d \\ \sqrt{mp}\,\|\overrightarrow{\mathbf{R}}_A \mathbf{H}_{x_i,\mathbf{y}^*}\|_\infty \leq \sqrt{mp}\,\ell m \cdot \|\mathbf{H}_{x_i,\mathbf{y}^*}\|_\infty \leq \sqrt{p} \cdot \ell 2^d m^{1.5+d} \leq \tau_s. \end{cases}$$

Then, having $\mathbf{R}_{x_i}$, the challenger can compute $\mathsf{sk}_{x_i,t}$ using ABS.KeyGen and ABS.KeyUpdate algorithms.

$t > t^*$**:** In this case, the condition $U_{x_i}(\mathbf{y}^*) = C^*(x_i) = 1$ is not necessarily verified. Indeed, because of the forward-security, an attacker can ask a key for (C^*, M^*), for a time period $t > t^*$, and therefore $C^*(x_i) = 0$, in which case we can not compute the keys as previously. Then, for each node $z \in \mathrm{Nodes}_{(t \to T-1)}$, the challenger first computes the smallest index $d_{z,t}$ such that $1 \leq d_{z,t} \leq d$ and $\mathsf{Bin}(t^*)[d_{z,t}] \neq z[d_{z,t}]$.

Then he computes $\mathsf{sk}_{x_i}[z] = \mathrm{RandBasis}(\mathrm{ExtBasis}(\mathbf{S}_{d_{z,t}}, \mathbf{A}_{x_i,z}), s_{d_{z,t}})$. Finally, he sets $\mathsf{sk}_{x_i,t}$ as in the ABS scheme and sends it to the adversary.

The distribution of $\mathsf{sk}_{x_i,t}$ remains the same in both cases. Note that in the second case, when $t > t^*$, RandBasis algorithm outputs bases that are statistically close when taking as input two different bases of the same lattice. Thus, Game 1 and Game 2 are statistically indistinguishable.

Game 3**:** Finally, we change the way the challenger samples $\mathbf{A}$. Instead of sampling it with a trapdoor as in ABS.Setup, he simply samples an uniform $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, so that the distribution is close to the one in the previous game and the two games are statistically indistinguishable.

Then, we replace the challenger in Game 3 with an algorithm $\mathcal{B}$ for which we give the description below and which solves the SIS problem using a forged signature generated by the adversary.

Algorithm $\mathcal{B}$ is given $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ uniform and then he plays the forward-secure no-signing-query unforgeability game with $\mathcal{A}$ using matrix $\mathbf{A}$.

Assume that $\mathcal{A}$ produces a valid forgery $\sigma_{\mathbf{y}^*,t^*}$ for $(\mathbf{y}^* = \mathsf{bitstr}(M^*, C^*), t^*)$ given at the beginning of the game. Then $\sigma_{\mathbf{y}^*,t^*} \neq \mathbf{0}$, $\|\sigma_{\mathbf{y}^*,t^*}\|_\infty \leqslant B$ and $[\mathbf{A}\|\overrightarrow{\mathbf{A}} - \mathbf{y}^* \otimes \mathbf{G}\|\mathbf{T}_1^{\mathsf{Bin}(t^*)[1]}\|\cdots\|\mathbf{T}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*} = \mathbf{0}$, and $[\mathbf{A}\|\mathbf{A}\overrightarrow{\mathbf{R}}_A\|\mathbf{A}\mathbf{R}_1^{\mathsf{Bin}(t^*)[1]}\|\cdots\|\mathbf{A}\mathbf{R}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*} = \mathbf{A} \cdot [\mathbf{I}\|\overrightarrow{\mathbf{R}}_A\|\mathbf{R}_1^{\mathsf{Bin}(t^*)[1]}\|...\|\mathbf{R}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*} = \mathbf{0}$. Since $\|[\mathbf{I}\|\overrightarrow{\mathbf{R}}_A\|\mathbf{R}_1^{\mathsf{Bin}(t^*)[1]}\|...\|\mathbf{R}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*}\|_\infty \leqslant (\ell(m+d)+1)\|\sigma_{\mathbf{y}^*,t^*}\|_\infty = (\ell(m+d)+1)B \leq B'$, it means that $[\mathbf{I}\|\overrightarrow{\mathbf{R}}_A\|\mathbf{R}_1^{\mathsf{Bin}(t^*)[1]}\|...\|\mathbf{R}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*}$ is a valid solution for $\mathrm{SIS}_{n,q,B',m'}$. □

We focus now on the security proofs of our forward secure group signature scheme, including the correctness, the CCA-selfless anonymity and the traceability.

Correctness. The correctness of the GS scheme follows directly from the correctness of OTS, ABS and SKE. We prove that a signature $\Sigma = (\mathsf{ct}, \mathsf{ovk}, \sigma, \tau) \leftarrow \mathbf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t)$ that was correctly generated passes the verification. We have $\mathsf{OTS.Verify}(\mathsf{ovk}, \tau, M\|\sigma) = \mathsf{Valid}$ by the correctness of OTS. Then, by the correctness of SKE we have $C[\mathsf{ovk}, \mathsf{ct}](i||K_i) = 0$ and therefore $\mathsf{ABS.Verify}(\mathsf{mpk}, C[\mathsf{ovk}, \mathsf{ct}], \sigma, M, t) = \mathsf{Valid}$.

Theorem 3 (**CCA-selfless anonymity**)**.** *If ABS is perfectly private and adaptive unforgeable, OTS is strongly unforgeable and SKE is IND-CCA secure and key-robust, then GS constructed as above is CCA-selfless anonymous.*

Theorem 4 (**Traceability**)**.** *If ABS is forward-secure (adaptively) unforgeable and SKE has key-robustness then the group signature scheme constructed above has the forward-secure traceability property.*

Proof. We fix an adversary $\mathcal{A}$ and we consider the traceability game that he is playing with a challenger. Let (M^*, Σ^*, t^*) be the forgery output by $\mathcal{A}$ which can be of either type:

— Type I forgeries are those for which the Open algorithm fails to identify the signer. We define E_1 to be the event that $\mathcal{A}$ wins the game and $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*) = \mathsf{Invalid}$ holds.

— Type II forgeries are those for which the Open algorithm traces to an uncorrupted member or to a corrupted member that requested keys for time periods after t^*. We define E_2 to be the event that $\mathcal{A}$ wins the game and $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*) = i^*$ such that $i^* \notin \mathcal{CU}$ or $i^* \in \mathcal{CU}$ but $\mathcal{A}$ did not query $\mathsf{gsk}_{i^*,t}$ for $t \leq t^*$.

We handle the two kind of forgeries in the following two lemmas.

Lemma 12. *If ABS is forward-secure adaptively unforgeable then* $Pr[E_1] = \mathrm{negl}(\lambda)$.

Proof. This is a proof by contradiction where we assume that E_1 happens with non-negligible probability ϵ and show how to construct an adversary $\mathcal{B}$ that breaks the adaptive unforgeability of ABS with the same probability.

The game begins when the challenger sends $\mathcal{B}$ the security parameter λ and public key mpk. Then $\mathcal{B}$ samples $\mathsf{pp} \xleftarrow{\$} \mathsf{SKE.Setup}(1^{\check{}})$, $K_i \xleftarrow{\$} \mathsf{SKE.Gen}(\mathsf{pp})$ for $i \in [N]$ and sets $\mathsf{gpk} = (\mathsf{pp},\mathsf{mpk})$, $\mathsf{gok} = \{K_i\}_{i\in[N]}$ and gives $(\mathsf{gpk}, \mathsf{gok})$ to $\mathcal{A}$. During the game, $\mathcal{B}$ answers to the queries of $\mathcal{A}$, for a certain period of time t, as follows:

— For a corrupt query for (i, t), $\mathcal{B}$ makes a key query $(i, x_i = i\|K_i, t)$ to its challenger who returns $\mathsf{sk}_{i\|K_i,t}$. Then $\mathcal{B}$ returns $\mathsf{gsk}_{i,t} = (i, K_i, \mathsf{sk}_{i\|K_i,t})$ to $\mathcal{A}$.

— To a signing query (i, M, t), $\mathcal{B}$ answers as follows: $\mathcal{B}$ samples $(\mathsf{ovk},\mathsf{osk}) \xleftarrow{\$} \mathsf{OTS.KeyGen}(1^\lambda)$, computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{K}_i, i||\mathsf{ovk})$ and makes a signing query $(C[\mathsf{ovk}, \mathsf{ct}], M, t, i)$ to its challenger who replies with $\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C[\mathsf{ovk}, \mathsf{ct}], M, t)$ after verifying that $C(i||K_i) = 0$. Then, $\mathcal{B}$ runs $\tau \leftarrow \mathsf{OTS.Sign}(\mathsf{osk}, M||\sigma)$ and returns $\Sigma = (\mathsf{ovk}, \mathsf{ct}, \sigma, \tau)$ to $\mathcal{A}$.

At the end, $\mathcal{A}$ will output a forgery $(M^*, \Sigma^* = (\mathsf{ovk}^*, \mathsf{ct}^*, \sigma^*, \tau^*), t^*)$. If $\mathsf{GS.Verify}(\mathsf{gpk}, M^*, \Sigma^*, t^*) = \mathsf{Valid}$ and $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*) = \mathsf{Invalid}$ do not hold then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ outputs $(M^*, C[\mathsf{ovk}^*, \mathsf{ct}^*], \sigma^*, t^*)$ as its forgery.

We show that $\mathcal{B}$ wins the game whenever E_1 happens. We note that $\mathsf{GS.Verify}(\mathsf{gpk},M^*, \Sigma^*, t^*) = \mathsf{Valid}$ implies $\mathsf{ABS.Verify}(\mathsf{mpk},C[\mathsf{ovk}^*, \mathsf{ct}^*], \sigma^*, t^*) = \mathsf{Valid}$. We need to show $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ for all $i \in N$ for which $\mathcal{B}$ has made a corrupt query for time $t \leq t^*$. It is easy to see that since $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*) = \mathsf{Invalid}$ holds, there is no $i \in [N]$ so that $\mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}$. We immediately have that $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ for all $i \in N$ and any time t of the corrupt query. Since $\mathcal{B}$'s

simulation is perfect, we conclude that $\mathcal{B}$ wins the game with probability ϵ. This concludes the proof of the lemma.

□

Lemma 13. *If ABS is forward-secure adaptively unforgeable and SKE has key-robustness then* $Pr[E_2] = \mathsf{negl}(\lambda)$.

This is a proof by contradiction where we assume that E_2 happens with non-negligible probability ϵ and show how to construct an adversary $\mathcal{B}$ that breaks the forward-secure unforgeability of ABS with non-negligible probability. We consider the following sequence of games where F_i represents the probability that E_2 occurs and the challenger does not abort in Game i.

Game 0**:** The first game is the forward-secure traceability game between adversary $\mathcal{A}$ and challenger. Assume that $Pr[F_0] = \epsilon$.

Game 1**:** In this game, the challenger makes a guess for i^* as $j^* \xleftarrow{\$} [N]$ and for time t^* as $z^* \xleftarrow{\$} [T]$ at the beginning of the game and aborts if $j^* \neq i^*$ and $z^* \neq t^*$ at the end of the game. Note that the view of $\mathcal{A}$ is independent from j^* and GS.Open outputs only two possible symbols: an integer $i \in [N]$ and Invalid, therefore we have $Pr[F_1] = \epsilon/(N * T)$.

Game 2**:** Here, the challenger aborts the game when his guess (j^*, z^*) turns out to be false (meaning that $j^* \neq i^*$ or $z^* \neq t^*$) . This can happen when $\mathcal{A}$ makes a corruption query for j^* at time $t \leqslant z^*$ or i^*, t^* defined at the end of the game do not equal to j^*, z^* respectively. This change does not have any effect on the probability, therefore $Pr[F_1] = Pr[F_2]$.

Game 3**:** In this game, the challenger aborts at the end of the game if $|\{i \in [N]$: $\mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\ \}| \neq 1$ for the forgery $(M^*, \Sigma^* = (\mathsf{ovk}^*, \mathsf{ct}^*, \sigma^*, \tau^*), t^*)$ output by $\mathcal{A}$ at the end of the game. We show that the probability to have E_2 and $|\{i \in [N]$: $\mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\ \}| \neq 1$ happening at the same time is negligibly small. We note that E_2 implies that $\mathsf{SKE.Dec}(K_{i^*}, \mathsf{ct}^*) \neq \mathsf{Invalid}$ for $i^* \in [N]$ and together with the abort condition results in $|\{i \in [N]$: $\mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\ \}| \geq 2$. Further we show that the probability of having the last inequality is negligible.

$Pr[|\{i \in [N]\text{: } \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\ \}| \geq 2]$

$$\leq Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{SKE.Setup}(1^\lambda), \mathsf{K}_j \xleftarrow{\$} \mathsf{SKE.Gen(pp)} \text{ for } j \in [N]\text{:} \\ \exists \mathsf{ct}^*, \exists i, i^* \in [N] \\ s.t.\ i \neq i^*, \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}, \mathsf{SKE.Dec}(K_{i^*}, \mathsf{ct}^*) \neq \mathsf{Invalid} \end{array}\right]$$

$$\leq \sum_{i,i^* \in [N] s.t. i \neq i^*} Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{SKE.Setup}(1^\lambda), \mathsf{K}_i, \mathsf{K}_i^* \xleftarrow{\$} \mathsf{SKE.Gen(pp)}\text{:} \\ \exists \mathsf{ct}^*, \exists i, i^* \in [N] s.t. i \neq i^*, \\ \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}, \mathsf{SKE.Dec}(K_{i^*}, \mathsf{ct}^*) \neq \mathsf{Valid} \end{array}\right]$$

$$\leq N(N-1)/2 \cdot \mathrm{negl}(\lambda) = \mathrm{negl}(\lambda).$$

where the second inequality is by the union bound and the third one is by the key-robustness of SKE. Therefore $|Pr[F_2] - Pr[F_3]| = \mathrm{negl}(\lambda)$.

We then replace the challenger in Game 3 with an adversary $\mathcal{B}$ against the forward-secure unforgeability of ABS with advantage $Pr[E_3]$.

First, the challenger sends $\mathcal{B}$ the security parameter λ and public key mpk. Then $\mathcal{B}$ samples $\mathsf{pp} \xleftarrow{\$} \mathsf{SKE.Setup}(1^{\check{}})$, $K_i \xleftarrow{\$} \mathsf{SKE.Gen(pp)}$ for $i \in [N]$ and sets $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk})$, $\mathsf{gok} = \{K_i\}_{i \in [N]}$ and gives $(\mathsf{gpk}, \mathsf{gok})$ to $\mathcal{A}$. During the game, $\mathcal{B}$ answers to the queries of $\mathcal{A}$ as follows for a certain period of time t.

— For a corrupt query for (i, t): If $i = j^*$ and $t \leq z^*$, $\mathcal{B}$ aborts. Else, he makes a key query $(i, x_i = i||K_i, t)$ to its challenger who returns $\mathsf{sk}_{x_i,t}$. Then $\mathcal{B}$ returns $\mathsf{gsk}_{i,t} = (i, K_i, \mathsf{sk}_{x_i,t})$ to $\mathcal{A}$.

— To a signing query (i, M, t), $\mathcal{B}$ answers as follows: $\mathcal{B}$ samples $(\mathsf{ovk}, \mathsf{osk}) \xleftarrow{\$} \mathsf{OTS.KeyGen}(1^\lambda)$, computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{K}_i, i||\mathsf{ovk})$ and makes a signing query $(C[\mathsf{ovk}, \mathsf{ct}], M, t, i)$ to its challenger who replies with $\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C[\mathsf{ovk}, \mathsf{ct}], M, t)$ after verifying that $C(i||K_i) = 0$. Then, it runs $\tau \leftarrow \mathsf{OTS.Sign}(\mathsf{osk}, M||\sigma)$ and returns $\Sigma = (\mathsf{ovk}, \mathsf{ct}, \sigma, \tau)$ to $\mathcal{A}$.

At the end, $\mathcal{A}$ will output a forgery $(M^*, \Sigma^* = (\mathsf{ovk}^*, \mathsf{ct}^*, \sigma^*, \tau^*), t^*)$. In this case, there are two situations where $\mathcal{B}$ aborts: if either $\mathsf{GS.Verify}(\mathsf{gpk}, M^*, \Sigma^*, t^*) = \mathsf{Valid}$ or $i^* = j^*$ or $t^* = z^*$ does not hold where $i^* = \mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*)$. It also aborts if $|\{i \in [N]\text{: } \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\ \}| \neq 1$. Otherwise, $\mathcal{B}$ outputs $(M^*, C[\mathsf{ovk}^*, \mathsf{ct}^*], \sigma^*)$ as its forgery.

In the following we show that $\mathcal{B}$ wins the game whenever event F_3 occurs. We verify that the conditions for the winner of the forward-secure unforgeability game are satisfied. Since $\mathsf{GS.Verify}(\mathsf{gpk}, M^*, \Sigma^*, t^*) = \mathsf{Valid}$ we have $\mathsf{ABS.Verify}(\mathsf{mpk}, C[\mathsf{ovk}^*, \mathsf{ct}^*], \sigma^*, M, t^*) = \mathsf{Valid}$. We then show that $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ for any key queried by $\mathcal{B}$ respective to x_i with $i \in [N] \setminus \{i^*\}$ and t where $t \leq t^*$. We note that $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ is true since $\mathsf{SKE.Dec}(\mathsf{K}_i^*, \mathsf{ct}^*) \neq \mathsf{Invalid}$ and $|\{i \in [N]\colon \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\}| = 1$. Moreover, $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ for any time period since the secret keys K_i involved in the circuit do not depend on the time period.

It remains to show that $\mathcal{B}$ has never made signing queries for $(M^*, C[\mathsf{ovk}^*, \mathsf{ct}^*], t^*)$. Because $\mathcal{A}$ has won the game, we have $M^* \neq M$ which implies $(M^*, C[\mathsf{ovk}^*, \mathsf{ct}^*], t^*) \neq (M, C[\mathsf{ovk}, \mathsf{ct}], t)$. Therefore we have that the winning probability of $\mathcal{B}$ is exactly $Pr[F_3]$. This concludes the proof of the lemma.

□

This concludes the proof of the theorem since it follows that the advantage of the adversary $\mathcal{A}$ in the traceability game is negligible.

□

Chapter 5

Lattice-based blind signature without restarts

We present in this chapter, a blind signature and its partially blind variant based on lattices assumptions. In a nutshell, the blind signature scheme allows any user, who wants to sign a message of its choice, to interact with a signer, who possesses the signing key. Blind signature is a cornerstone in privacy-oriented cryptography and we propose here the first lattice-based scheme without restart that is based on the former lattice-based blind signature of Rückert [Rüc10]. In 2020, Hauck et al. published a paper in this topic at Crypto 2020 which raise a flaw in the security proof that appears in all previous paper and they propose a construction that solves this problem but with a big efficiency cost. In comparison our construction is more efficient to the one of [Hau+20], but our security relies on a conjecture.

Compare to related work, the key idea of our construction is to provide a trapdoor to the signer in order to let him perform some gaussian pre-sampling during the signature generation process, preventing this way to restart from scratch the whole protocol. The security of our scheme relies on the ring k-SIS assumption, in the random oracle model. We also explain the security issue raised by [Hau+20] in lattice-based blind signature one-more unforgeability proof and how we partially overcome it. In fact this issue is inherited from the seminal work of Rückert [Rüc10] and induced by a too direct adaptation of the former one-more unforgeability proof from Pointcheval and Stern [PS00]. We finally present a partially blind variant of our scheme, which is done with no supplementary cost, as the number of elements generated and exchanged during the signing protocol is exactly the same.

We start by giving the generic definition of a blind signature scheme and the various properties, security included, that it must fulfil. We then explain the generic framework from Rückert [Rüc10] that is used by all constructions, including our, in the lattice settings. We continue by developing our construction in detail, together with its partially

blind variant. We end this chapter by proving the security of our scheme, especially the one-more unforgability proof that is proved under some conjecture, in which we give a complete analysis.

5.1 Blind signature definition

Blind signatures, introduced by Chaum in 1982 [Cha82], permit a user to obtain a signature on a chosen message by interacting with a signing authority. They have recently been standardized at ISO/IEC 18370 and are deployed in e.g., the Microsoft U-Prove technology. The main difference with a classical signature is that at the end of the signature generation, the authority has never seen the message and is not able to later link the signature output by the user to its corresponding view of the interactions. Thus, the user is anonymous among the set of users having requesting a signature to this authority. Blind signatures are then usually used to provide anonymity in practical services such as e-vote or e-cash [Cha82]. In these cases, the authority provides the ability to vote or spend coins to a user, but the latter does not want his vote or payment to be traced. Pointcheval and Stern [PS96a], and then Juels et al. [JLO97], have proposed formal definitions for the security of blind signatures, namely *blindness* (the authority cannot make the link between a (message, signature) pair and its transcript of the signing protocol) and *one-more unforgeability* (a user cannot output more valid (message, signature) pairs than the number of times he has interacted with the authority).

5.1.1 Generic definition and additional properties

A blind signature scheme is an interactive protocol between two parts. The first part is a common user, who wants to obtain a signature on a message of its choice, but with no access to the signing key. Then he has to interact with an authority in possession of this key to generate the signature, these interactions are composed of at least 2 exchanges (for example blind RSA [Fer93]) between the user and the signer. The first interaction is the sending of a challenge by the user and the second interaction is the answer with the signature corresponding to the received challenge, from the signer. Indeed, to get a blind signature, the user generates the challenge, that is included in the outputted signature, he must blind it into before to send it to the signer. It means the signer computes a signature on the challenge, that has been blinded, and sends it to the user, who modifies

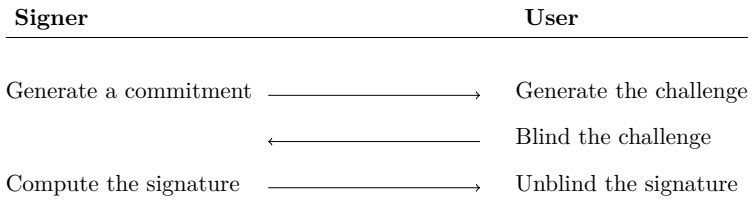

Figure 5.1 – 3-move blind signature.

the signature accordingly to fit with the outputted challenge. In the rest of this chapter, we call this last step the "unblind" operation of the signature. Thus the signer has never seen the genuine challenge and is unable to trace back the blind signatures.

In some situations, these 2 interactions are not enough to compose a blind signature scheme and a third interaction, which appears at the beginning of the protocol, is necessary. In fact this 3 exchanges scenario is copied from the zero-knowledge proofs. In this 3 exchange interaction, the signer starts the protocol by sending a commitment, that will be used by the user to help him to generate its challenge and perform the rest of the protocol.

Now, before to take a closer look to the formal definitions and security properties we can expect from a blind signature, we define a variant of such tool. Coming back to the e-voting and e-cash use cases, the basic blind signature is sometimes too restrained to be used. Indeed considering an e-cash scenario, where a blind signature is associated to an electronic coin, we would like to add a validity date or a monetary value that should be made public. This is why we need a slightly modified version, where the message to be signed can include some information, that the authority should know. This variant of blind signatures, called *partially* blind signatures, has been introduced in [AF96]. Concretely, it means that the two parties agree on a common and public information added to the signed message during the blind signature process. Obviously, this common information should not permit to break the security properties considered in a blind signature scheme, that should be modified accordingly.

We now give more formal definitions for blind and partially blind signature. We develop them on the protocol side as well as on the security side.

A (Partially) Blind Signature scheme (BS or PBS) consists of three algorithms (Keygen, Sign, Verif), where Sign is an interactive protocol between a signer $\mathcal{S}$ and a user $\mathcal{U}$. There are different ways to describe such an interactive protocol. In this work, we consider a three-move protocol as in [Oka92; Rüc10], as shown in Figure 5.1.

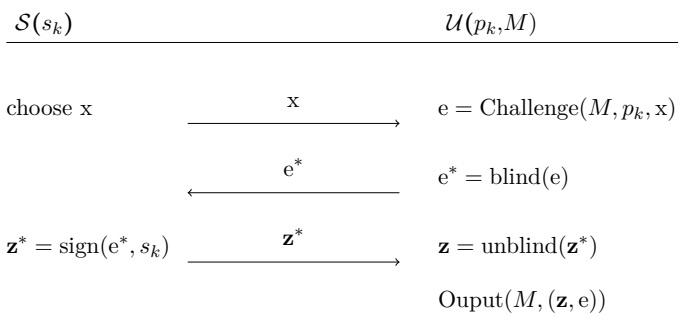

Figure 5.2 – Signing protocol.

— *Key Generation.* Keygen(1^n) given the security parameter n, outputs a private signing key s_k and a public verification key p_k.

— *Signature Protocol.* The interaction between the signer $\mathcal{S}(s_k)$ and the user $\mathcal{U}(p_k, M)$ is described in Figure 5.2. The input of $\mathcal{S}$ is a secret key s_k and the input of the user is the public key p_k and a message $M \in \mathcal{M}$, where $\mathcal{M}$ is the message space. The output of $\mathcal{S}$ is a transcript $(\mathrm{x}, \mathrm{e}^*, \mathbf{z}^*)$, with the internal values, of the signature generation and the output of $\mathcal{U}$ is a signature $\mathbf{z}$ on the message M, under s_k with respect to the challenge e.

In the case of partially blind signature, an additional information info is commonly chosen by the signer and the user. This common information info is public and is added to the message to be signed during the process.

— *Signature Verification.* The algorithm Verif$(p_k, M, (\mathbf{z}, \mathrm{e}))$ outputs 1 if $\mathbf{z}$ is a valid signature on M under p_k with respect to the challenge e, otherwise it outputs 0. In the case of the partially blind signature, the common information info is also needed as an input of the signature verification process.

Completeness. We define the completeness as in a digital signature scheme, i.e., for each honestly created signature with honestly created keys and for any message $M \in \mathcal{M}$ (and info), the signature verification has to be valid under these elements.

The security of (partially) blind signature schemes is then composed of two properties: (partial) blindness and one-more unforgeability, developed in the works of [JLO97; PS00].

(Partial) blindness. The first property a (partially) blind signature scheme must fulfil is the (partial) blindness property. It informally means that the signer is unable to link a valid (partially) blind signature $(M, (\mathbf{z}, \mathrm{e}))$ to the transcript $(\mathrm{x}, \mathrm{e}^*, \mathbf{z}^*)$ which generated it.

$\mathrm{Exp}^{\mathrm{blind}}_{\mathcal{S}^*,\mathrm{BS}}(n)$

$b \leftarrow_\$ \{0,1\}$
$(p_k, s_k) \leftarrow_\$ \mathrm{BS.Keygen}(1^n)$
$(M_0, M_1, \mathsf{info}_0, \mathsf{info}_1, \mathsf{state}_{\mathsf{find}}) \leftarrow_\$ \mathcal{S}^*(\mathrm{find}, p_k, s_k)$
$\mathrm{state}_{\mathsf{issue}} \leftarrow_\$ \mathcal{S}^{*\langle\cdot,\mathcal{U}(p_k,M_b,\mathsf{info}_b)\rangle,\langle\cdot,\mathcal{U}(p_k,M_{1-b},\mathsf{info}_{1-b})\rangle}(\mathsf{issue}, \mathsf{state}_{\mathsf{find}})$
Let $\mathbf{z}_b$ and $\mathbf{z}_{1-b}$ be the outputs of $\mathcal{U}(p_k, M_b, \mathsf{info}_b)$ and $\mathcal{U}(p_k, M_{1-b}, \mathsf{info}_{1-b})$, respectively.
If $\mathbf{z}_0 \neq$ fail and $\mathbf{z}_1 \neq$ fail and $\mathsf{info}_0 = \mathsf{info}_1$
  $d \leftarrow_\$ \mathcal{S}^*(\mathsf{guess}, \mathbf{z}_0, \mathbf{z}_1, \mathrm{state}_{\mathsf{issue}})$
Else
  $d \leftarrow_\$ \mathcal{S}^*(\mathsf{guess}, \mathsf{fail}, \mathsf{fail}, \mathrm{state}_{\mathsf{issue}})$
Return 1 iff $d = b$

$\mathrm{Exp}^{\mathrm{omf}}_{\mathcal{U}^*,\mathrm{BS}}(n)$

$\mathrm{H} \leftarrow_\$ \mathcal{H}(1^n)$
$(p_k, s_k) \leftarrow_\$ \mathrm{BS.Keygen}(1^n)$
$\{(M_1, (\mathbf{z}_1, \mathrm{e}_1)), ..., (M_m, (\mathbf{z}_m, \mathrm{e}_m)\} \leftarrow_\$ \mathcal{U}^{*\langle\mathcal{S}(s_k,\mathsf{info}),\cdot\rangle^\infty}(p_k)$
Let l be the number of successful interaction between $\mathcal{U}^*$ and the signer.
Return 1 iff
  $1. M_i \neq M_j$ for all $1 \leq i < j \leq m$
  $2. \mathrm{BS.Verif}(p_k, M_i, (\mathbf{z}_i, \mathrm{e}_i), \mathsf{info}) = 1$ for all $i = 1, ..., m$
  $3. m = l + 1$.

Figure 5.3 – Security experiments

Definition 16. ***(Partial) Blindness*** *is formalized in the experiment* $\mathrm{Exp}^{\mathrm{blind}}_{\mathcal{S}^*,\mathrm{BS}}(n)$ *in Fig. 5.3. A* BS *scheme is* (t, δ)*-blind, if there is no adversary* $\mathcal{S}^*$*, running in time at most* t*, that wins the blindness experiment with advantage at least* δ*, where the advantage is defined as* $\mathrm{Adv}^{\mathrm{blind}}_{\mathcal{S}^*,\mathrm{BS}} = |Prob[\mathrm{Exp}^{\mathrm{blind}}_{\mathcal{S}^*,\mathrm{BS}}(n) = 1] - \frac{1}{2}|$.

Concerning this experiment, we consider an adversary, acting as a signer $\mathcal{S}^*$, that works in three modes. In mode find, the adversary chooses two messages M_0, M_1 (and corresponding common informations info_0 and info_1). Then, in mode issue, he interacts with two users. Each user gets the two messages M_0 and M_1 that they divide between them and, on a coin flip b, each user interacts with the adversary $\mathcal{S}^*$, as in the signing protocol, to generate a blind signature $\mathbf{z}_b$ (resp $\mathbf{z}_{1-b}$) for the message M_b (resp. M_{1-b}). After seeing both unblinded signatures $\mathbf{z}_0, \mathbf{z}_1$ in the original order, with respect to M_0 and M_1, the signer enters the third mode guess and has to guess the bit b of the corresponding signatures. If any of the signature process fails, the signer only gets a notification of failure. The adversary is moreover allowed to keep a state that is fed back in subsequent calls.

The blindness experiment is available for both blind and partially blind signatures. The only difference between the two cases is that we have to deal with the common information info in the partially blind version. It means that, in this case we have $\mathsf{info}_0 = \mathsf{info}_1 (= 0)$. One has just to remove the variables info_0 and info_1 to obtain the exact experiment for basic blind signatures.

One-more unforgeability. The second security property is the one-more unforgeability. Informally, it means that a user, given l valid signatures generated following l interactions with a signer, can not output $(l+1)$ valid signatures. The one-more unforgeability property ensures that each completed interaction between a signer and a user provides at most one signature. In the corresponding experiment, an adversarial user tries to output m valid signatures after $l < m$ completed interactions with an honest signer.

Definition 17. ***One-more unforgeability*** *is formalized in Fig. 5.3. A blind signature scheme* BS *is* $(t, q_{\text{Sign}}, \delta)$*-one-more unforgeable if there is no adversary* $\mathcal{A}$*, running in time at most* t*, making at most* q_{Sign} *signature queries, who wins the one-more unforgeability experiment with probability at least* δ*.*

As for the blindness property, this experiment is available for both blind and partially blind signatures. Indeed, one can adapt the given experiment to the case of basic blind signatures by simply removing the variables info. The difference between the two variants is that in the partially variant we also need to verify that the attacker can not forge a signature on an information never appearing in the l previous interactions with the signer.

5.1.2 Blind signature and lattices

Now that we have defined what is a blind signature and what property we can expect for such constructions, we will take a closer look on the particular case of lattice-based blind signature and study how the line of research on the subject progressed from the first scheme that has been designed by Rückert in 2010 [Rüc10], upon the idea of the Okamoto-Schnorr blind signature scheme [Oka92], to the very recent one by Hauck et al. in 2020 [Hau+20].

In fact, before the seminal paper of [Rüc10] inspiring all its successor, the history of lattice-based blind signature begins with the identification scheme of Lyubashevsky [Lyu08]. Indeed, from an identification scheme there is a generic transformation to get a blind signature, by using a transformation that follows the same spirit than the Fiat-Shamir transformation [FS86] as originally applied on the Schnorr identification scheme [Sch89] to get the Okamoto-Schnorr blind signature scheme [Oka92].

Identification scheme and blind signatures. In an identification scheme, a prover and a verifier interact such that at the end of the protocol, the verifier is convinced of the identity of the prover. In practice, the prover possesses a secret key corresponding to a

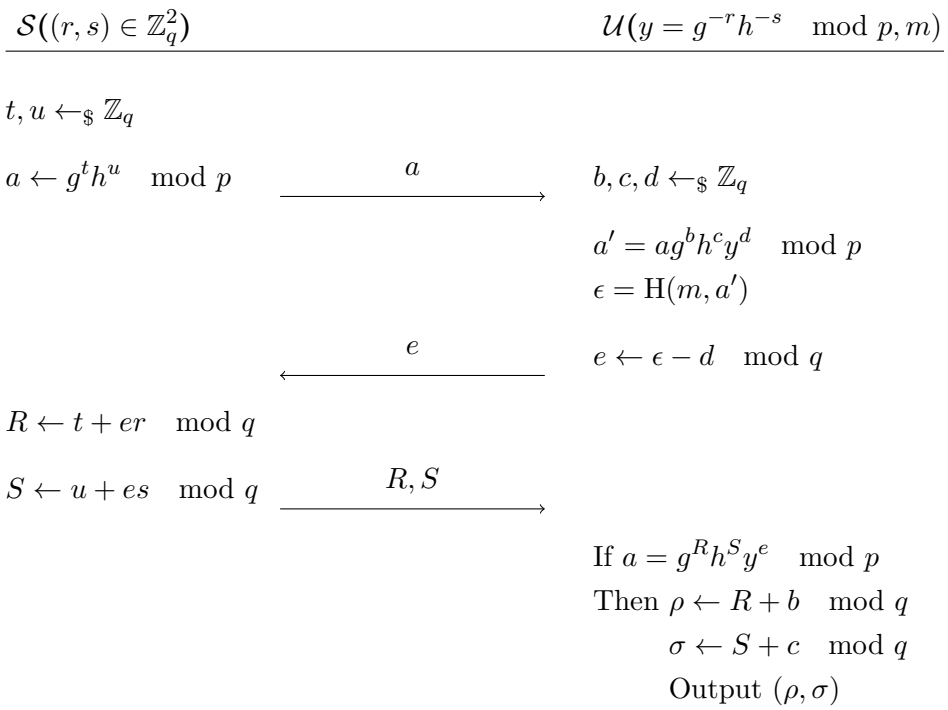

Figure 5.4 – Okamoto-Schnorr blind signature scheme [Oka92]

public key that is known by the verifier. Then the prover tries to convince the verifier that he knows the secret key in a zero-knowledge way, meaning that the verifier gets no other information other than the fact that the prover is indeed, in possession of the claimed secret key.

As a concrete example, we can cite the identification scheme of Schnorr [Sch89] that was turned into a blind signature scheme by Okamoto [Oka92] to give birth to the Okamoto-Schnorr blind signature scheme, described in Figure 5.4. The parameters are two prime integers p and q such that $q|(p-1)$, while g and h are elements of $\mathbb{Z}_p^\star$ of order q and H is a hash function modeled as a random oracle.

The idea of the Schnorr identification scheme was applied on the identification scheme of Lyubashevsky [Lyu08] that we recall below, the parameters are $p = \tilde{O}(n^3), m = \lceil 4n \log n \rceil$ and the SAFE set is equal to $\{1, \ldots, 5m-1\}^m$.

In the Lyubashevsky scheme (Figure 5.5), a prover succeeds to prove its identity if the verifier is convinced that he owns a short vector $\tilde{\mathbf{w}} \in \{0,1\}^m$ corresponding to a published hash value $\mathbf{w} = \mathbf{A} \cdot \tilde{\mathbf{w}} \mod p$. So now the prover wants the ability to prove to any verifier that he is in possession of such a vector $\tilde{\mathbf{w}}$ verifying the above relation. An easy way would be to reveal the value $\tilde{\mathbf{w}}$ such that anyone can compute the correct $\mathbf{w}$ and valid his

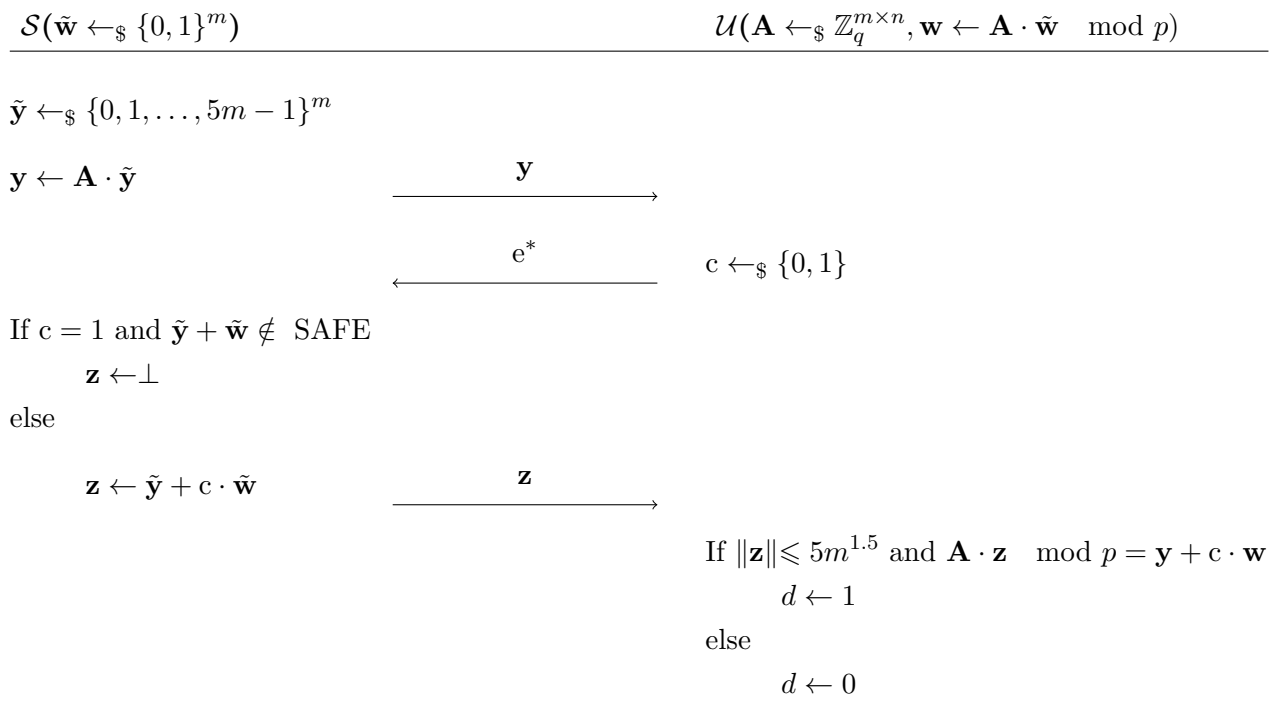

Figure 5.5 – Identification scheme of Lyubashevsky [Lyu08]

identification, but then everybody can steal its identity, and prove it in the same way.

In order to keep its vector $\tilde{\mathbf{w}}$ private, the idea is that the prover "masks" its secret vector $\tilde{\mathbf{w}}$, using the identification zero-knowledge property, with an ephemeral vector $\tilde{\mathbf{y}}$ that is generated during the execution of the protocol, and which is different for each iteration. Finally the prover reveals the vector $\mathbf{z} = \tilde{\mathbf{y}} + \mathrm{c} \cdot \tilde{\mathbf{w}}$ to the verifier, who check the relation $\mathbf{A} \cdot \mathbf{z} \mod p = \mathbf{y} + \mathrm{c} \cdot \mathbf{w}$ to validate the identification. The problem now is that the distribution of the vector $\mathbf{z}$ is not a leak-free distribution, indeed if the challenge is $\mathrm{c} = 1$, then the vector $\mathbf{z}$ is centered on $\tilde{\mathbf{w}}$ and not on 0, giving by the way some informations about the secret $\tilde{\mathbf{w}}$.

Then to ensure that the vector $\mathbf{z}$ gives no information about the secret $\tilde{\mathbf{w}}$, the protocol accept it or reject it following the value it takes, using the well-known rejection sampling technique, introduced in [Lyu08]. Concretely, a set labeled as "SAFE" is considered, such that we know that if the vector $\mathbf{z}$ falls into that set, it means that it reveals no information on the secret $\tilde{\mathbf{w}}$. The protocol stops and is repeated each time the vector $\mathbf{z}$ is not in the set "SAFE", while it reaches its end as soon as a vector $\mathbf{z}$ is generated by the prover and belongs to the set "SAFE": here comes the name of rejection sampling.

Starting from this identification scheme, the idea of Rückert [Rüc10], was to apply to the lattice settings a transformation equivalent to the one that Okamoto carried out on the discrete-logarithm based Schnorr construction, creating the so-called Okamoto-Schnorr scheme. According to the Fiat-Shamir transform, the idea is that the challenge is not anymore a single bit compared to the identification scheme, but instead we consider the output of a random oracle, that can take at least 2^n possible values. However, since the challenge can not be sent in clear, to preserve the blindness, the output of the random oracle must be blinded before to be sent to the signer. Using all these tricks, Rückert [Rüc10] managed to build the first lattice-based blind signature scheme, construction on which we based our own work and that we explain further.

5.1.3 Technical details of the Rückert scheme

Ruckert scheme. We now describe the Rückert protocol, we start by defining some elements that are inherent to its construction as well as our own construction. First the scheme is built on the lattice polynomial variant R_q to benefit of its efficiency compared to $\mathbb{Z}_q$. On the R_q ring, we can use the hash function family described in 3.3.4.

In fact, the reduction of the one-more unforgeability property is brought back to the problem of finding a collision on a hash function that belongs to this family, and that can be seen as a solution to the Ring-SIS problem. Concerning the function denoted H, it is seen as a random oracle and can be the function SHA-256 for example.

We also note that the sets $D_y, D_\alpha, \ldots$ are some integers intervals on which the variables are uniformly picked. We emphasize that this notation is restricted to the Rückert scheme and denotes a gaussian distribution everywhere else. Finally two rejection sampling, one on the signer side and one on the user side, are represented in the same way as in the identification scheme of [Lyu08] with G_*^m and G^m that identifies in the same way as the set "SAFE" in [Lyu08].

The general framework of the blind signature of [Rüc10] (see Figure 5.6) is the same as the one from the identification scheme of Lyubashevsky [Lyu08], with the initial commitment as first step. At the same time, the signing step is very similar to the second step of the prover in [Lyu08], with only the distributions differing. However the challenge generation deeply differs to fit into the requirements of a blind signature as explained previously. After its generation, the challenge is blinded by adding a "mask", seen as a vector, on the original challenge, and by using the rejection sampling technique to erase any information that could leak. Finally to retrieve the signature corresponding to the

Signer(s)		**User(S,M)**
$\tilde{\mathbf{y}} \leftarrow_{\$} D_y^m$		
$\mathrm{y} \leftarrow h_{\mathbf{a}}(\mathbf{y})$	$\xrightarrow{\mathrm{y}}$	$r \leftarrow_{\$} \{0,1\}^n$
		$C \leftarrow com(M, r)$
		$t_1 \leftarrow_{\$} D_\alpha$
		$\mathbf{t}_2 \leftarrow_{\$} D_\beta^m$
		$e \leftarrow \mathrm{H}(\mathrm{y} - \mathrm{S}t_1 - h_{\mathbf{a}}(\mathbf{t}_2), C)$
		$e^* \leftarrow e - t_1$
		If $e^* \notin D_{\epsilon^*}$
$\mathbf{z}^* \leftarrow \mathbf{s}e^* + \tilde{\mathbf{y}}$	$\xleftarrow{e^*}$	Start over with fresh t_1
If $\mathbf{z}^* \notin G_*^m$		
Trigger restart	$\xrightarrow{\mathbf{z}^*}$	$\mathbf{z} \leftarrow \mathbf{z}^* - \mathbf{t}_2$
		If $\mathbf{z} \notin G^m$
		$result \leftarrow (C, t_1, \mathbf{t}_2, e)$
		Else
$Verification$	$\xleftarrow{result}$	$result \leftarrow ok$

Figure 5.6 – Rückert's blind signature scheme [Lyu08]

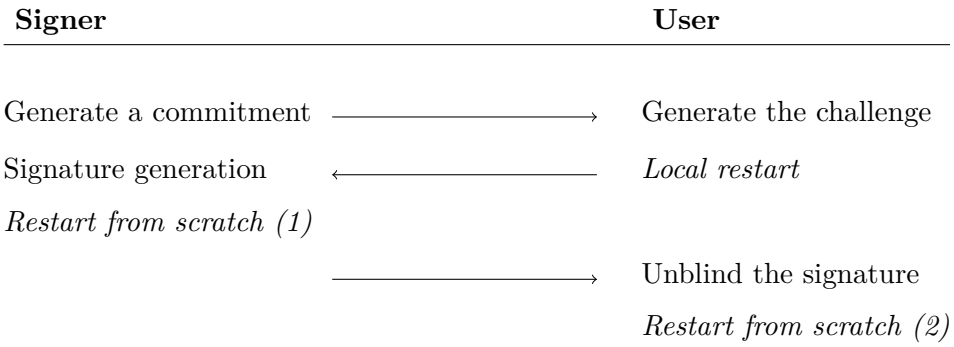

Figure 5.7 – Restarts in Rückert scheme.

original challenge, the user "unmasks" the signature with the second vector, by using once again a rejection sampling to preserve the final distribution of the signature from any leak of information.

Finally, we can see that Rückert introduces a final interaction between the signer and the user, in case that the final rejection sampling fails. It comes from the fact that the user can get an invalid blind signature, but then needs to notify it to the signer in order to start over the whole protocol. But to prevent the user to cheat and trigger a restart while he gets a valid blind signature, he needs to send all the elements that can prove to the signer that the signature is indeed invalid and that a restart is mandatory.

We can then see that the main restriction of the Rückert scheme is the solution to the correctness problems, solved by its restarts (see Figure 5.7). More precisely there is one *local restart* on the user side in its first step that is not a main problem with respect to efficiency. However the two *restarts from scratch* happening in the second step of the signer and of the user side are a lot more bothering, since they imply a restart of the whole protocol and not just the current step. These problems are incurred by the correctness issue brought by the rejection sampling, since each time a test of rejection sampling fails, it means that some elements need to be regenerated to give another chance to the rejection sampling to succeed. While the efficiency is affected by those restarts in an evident way, a more intricate issue appears in the security. We take a closer look into these issues in what follows next.

For a long time, the only existing blind signature on lattices was the scheme of Rückert [Rüc10], but the 4 several years have seen some resurgence of interest for the subject with the NIST competition started in 2016. Beginning with some improvements on the distribution of the elements composing the protocol, by switching from uniform to gaussian

distribution, done by Zhang et al [Zha+18]. Then, a new scheme called BLAZE [ABB19] (soon followed by its new version BLAZE+ [ABB20]) has been published, improving the user part of the original scheme of Rückert, by introducing a new and more efficient technique to generate the challenge as well as a technique using a Merkle tree to reduce the restarts. We can also denote two partially blind signatures, that has been designed by Tian et al [TZW16], later followed by [Pap+19]. Both schemes follows the transformation of Abe and Okamoto [AO00] to transform a blind signature into a partially blind variant.

More recently, a paper from Hauck et al. [Hau+20] revisited the lattice-based blind signature constructions. Indeed, they found a flaw into the seminal paper of Rückert, more precisely into its one-more unforgeability proof, that spreads and invalidates all blind signature existing constructions [TZW16; Zha+18; Pap+19; ABB19; ABB20], since they all rely on the Rückert proof for their one-more unforgeability security. The problem comes from the fact that there are some tricky issues when transforming the proof that was originally designed for schemes based on discrete logarithm or factorization into schemes based on lattices. We develop extensively the issue and the solution brought by [Hau+20] as well as the adaptation we brought into our solution in the section 5.3.

5.2 Our blind signature scheme

The main goal of our construction is to be more efficient than previous constructions. In order to achieve our goal, we get rid of these restarts, that can be restricting on the efficiency side as well as on the security side.

5.2.1 Our construction

Our main objective is then to find more efficient alternatives to the problems such restarts are solving. We then use several tricks to reach our goal and design a more efficient blind signature scheme.

— Our first difference is to base our construction on Eurocrypt 2012 Lyubashevsky's signature scheme [Lyu12]. By replacing all the uniform sampling distributions by gaussian ones, we benefit better parameters and a more efficient rejection sampling, compared to the Rückert scheme. In particular, the *local restart* on the user side is more efficient in our design.

— An other point that we introduce, is to give to the signer a trapdoor in order

to let him perform some gaussian presampling, and by then to avoid the restart triggered in the signing step. We then make use of the ring version of the efficient trapdoor function due to [MP12; GM18], in order to sample preimages of the one-way function defined in [LM06] associated to a public vector $\mathbf{a}$. Instead of generating a new challenge or an ephemeral vector in case of error, as it is done in the *restart from scratch (1)* of Rückert scheme (see Figure 5.7), the signer can execute some gaussian pre-sampling by himself to efficiently output a signature where the secret key is always sufficiently hidden.

— Using then a result due to Goldwasser et al. [Gol+10] on statistical distances between gaussian distribution centered on 0 and gaussian distribution centered on a vector $\mathbf{v}$, we hide the signature sent by the signer by adding an oversized vector, which is generated by the user. This naturally hides the information that can later be used by the signer to recognize the output blind signature, since the final signature distribution does not depend on the signature output by the signer. The consequence is that we do not need anymore the *restart from scratch (2)* in Figure 5.7, that has exactly the same objective.

— We remark that the removal of the trigger restarts leads to the uselessness of a commitment initially computed by the user during the challenge generation step. However the use of the trapdoor carries a problem in the one-more unforgeability proof, since the simulated signer needs to perform some gaussian pre-sampling but has no access to the trapdoor on the matrix $\mathbf{A}$. We address this problem using the k-SIS problem introduced in [BF11], and improved in [Lin+14], which allows the signer to get k short vectors of the kernel of $\mathbf{A}$ and asks him to output a $(k+1)$-th vector, linearly independent of the others.

We now detail the construction of our blind signature scheme $\mathrm{BS} = (\mathrm{KeyGen}, \mathrm{Sign}, \mathrm{Verif})$.

Setup. We consider the polynomial ring $\mathrm{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$, where the parameters q and n are expressed in Table 5.1. Two families of hash functions are necessary in the protocol, firstly a generic hash function $\mathrm{H} \leftarrow_{\$} \mathcal{H}(1^n){:}\{0,1\}^* \to \mathrm{R}_2$ (modelled as a random oracle), and a second one on the specific ring R_q, typically $h \leftarrow_{\$} \mathcal{H}(\mathrm{R}_q, m)$ as defined in the section 3.3.4 of the preliminaries.

The parameter table (Table 5.1) shows up the different sizes of the parameters involved in our blind signature scheme. The parameter n is chosen as a power of 2, in order to have the polynomial $X^n + 1$ irreducible and for efficiency reasons. The parameter m ensures

Parameter	Value	Asymptotic
n	power of 2	-
m	$\lfloor \log q \rfloor + 1$	$\Omega(\log n)$
γ	$n\alpha$	$\mathrm{O}(n\sqrt{n})$
α	$\omega(k\sqrt{\log n})$	$\mathrm{O}(\sqrt{n})$
β	$2^{\omega(\log n)}\sigma\sqrt{n}$	$\mathrm{O}(n^3\, 2^{\omega(\log n)})$
σ	$\omega((n\sqrt{n}\alpha)\sqrt{\log n})$	$\mathrm{O}(n^2\sqrt{n})$
D	$t\sqrt{n \cdot m}(\beta + \sigma)$	$\mathrm{O}(n^3\sqrt{n}\, 2^{\omega(\log n)})$
q	$\geqslant 4mn\sqrt{n}\log(n)D$.prime	$\Theta(n^6\, 2^{\omega(\log n)})$

Table 5.1 – Parameters of our scheme.

the worst-case to average case reduction of our scheme. The others parameters are set such that the different rejection sampling and security arguments work.

Key Generation. BS.Keygen(1^n) selects a secret key $\mathbf{s} \in \mathrm{R}_3^m$ and a vector of polynomial $\mathbf{a} = (\mathbf{a'}^T \| h\mathbf{g} - \mathbf{a'}^T\mathbf{T_a})^T \in \mathrm{R}_q^m$, along with a trapdoor $\mathbf{T_a}$ on $\mathbf{a}$, such that the hash function $h_\mathbf{a} \in \mathcal{H}(\mathrm{R}_q, m)$ is built with this polynomial vector $\mathbf{a}$. Finally the public key $\mathrm{p} = h_\mathbf{a}(\mathbf{s})$ is computed and made public. BS.Keygen(1^n) outputs $s_k = (\mathbf{s}, \mathbf{T_a})$ and $p_k = (\mathrm{p}, \mathbf{a})$.

Signature. BS.Sign(Signer$(\mathbf{s}, \mathbf{T_a})$, User$(\mathrm{p}, M)$) works as expressed in Figure 5.8. We recall that writing $\mathbf{v} \leftarrow$ PreSample$(\mathbf{T_a}, \mathrm{x}, \sigma)$ means that $h_\mathbf{a}(\mathbf{v}) = \mathrm{x} \bmod q$ and that $\mathbf{v}$ is following a Gaussian distribution of parameter σ. At the end of the protocol, the signer outputs a transcript, and the user outputs the uplet $(M, (\mathbf{z}, \mathrm{e}))$ composed by the message $M \in \{0.1\}^*$, the signature $\mathbf{z} \in \mathrm{R}^m$ and the challenge $\mathrm{e} \in \mathrm{R}_2$ to verify the signature.

Signer(s,$\mathbf{T_a}$)		**User(p,M)**
$\mathbf{y} \hookleftarrow D^m_{\mathrm{R},\sigma}$		
$\mathrm{x} \leftarrow h_{\mathbf{a}}(\mathbf{y})$	$\xrightarrow{\mathrm{x}}$	$\mathrm{t}_1 \hookleftarrow D_{\mathrm{R},\alpha}$
		$\mathbf{t}_2 \hookleftarrow D^m_{\mathrm{R},\beta}$
		If $\|\mathbf{t}_2\| > t\sqrt{n \cdot m} \cdot \beta$ then
		generate a fresh $\mathbf{t}_2$
		$\mathrm{e} \leftarrow \mathrm{H}(\mathrm{x} - \mathrm{p} \cdot \mathrm{t}_1 - h_{\mathbf{a}}(\mathbf{t}_2), M)$
		$\mathrm{e}^* \leftarrow \mathrm{e} - \mathrm{t}_1$
		Accept e^* with probability
		$\min(\frac{D_{\mathrm{R},\alpha}}{G_1 \cdot D_{\mathrm{R},\alpha,\mathrm{e}}}, 1)$
		Otherwise start over with a
$\mathbf{z}^* \leftarrow \mathrm{e}^* \cdot \mathbf{s} + \mathbf{y}$	$\xleftarrow{\mathrm{e}^*}$	fresh $\mathrm{t}_1 \hookleftarrow D_{\mathrm{R},\alpha}$
Accept $\mathbf{z}^*$ with probability		
$\min(\frac{D^m_{\mathrm{R},\sigma}}{G_2 \cdot D^m_{\mathrm{R},\sigma,\mathrm{e}^* \cdot \mathbf{s}}}, 1)$		
Otherwise :		
$\mathbf{z}^* \leftarrow \mathrm{PreSample}(\mathbf{T_a}, \mathrm{e}^* \cdot \mathrm{p} + \mathrm{x}, \sigma)$		
If $\|\mathbf{z}^*\| > t\sqrt{n \cdot m} \cdot \sigma$ then		
start over with a fresh		
$\mathbf{z}^* \leftarrow \mathrm{PreSample}(\mathbf{T_a}, \mathrm{e}^* \cdot \mathrm{p} + \mathrm{x}, \sigma)$	$\xrightarrow{\mathbf{z}^*}$	$\mathbf{z} \leftarrow \mathbf{z}^* - \mathbf{t}_2$
		Output $(M, (\mathbf{z}, \mathrm{e}))$

Figure 5.8 – BS protocol

Verification. The verification procedure $\mathrm{BS.Verif}(\mathrm{p}, M, (\mathbf{z}, \mathrm{e}))$ outputs 1 iff $\|\mathbf{z}\| \leqslant D$ and $\mathrm{H}(h_{\mathbf{a}}(\mathbf{z}) - \mathrm{p} \cdot \mathrm{e}, M) = \mathrm{e}$.

Now that we have explained our construction of lattice-based blind signature, we move into the security of our schemes. Firstly we investigate the property of blindness, that is quite straightforward thanks to the rejection sampling and the noise flooding techniques. We separate the one-more unforgeability in another section, since its history is really complicated and has been subject to several questioning in the last year.

5.2.2 Correctness and blindness

We start by proving the completeness property. It asks that for a genuinely generated blind signature, the verification algorithm outputs 1 with probability 1.

Theorem 5 (Completeness)**.** *The scheme* BS *is perfectly complete.*

Proof. Let us assume that the protocol outputs a valid signature. Then, for all honestly generated key pairs $(\mathbf{s}, \mathrm{p})$, all messages $M \in \{0,1\}^*$ and all signatures $(M, (\mathbf{z}, \mathrm{e}))$ we have $\|\mathbf{z}\| \leqslant \|\mathbf{z}^*\| + \|\mathbf{t}_2\| \leqslant D$ and:

$$\begin{aligned} h_{\mathbf{a}}(\mathbf{z}) - \mathrm{p}\cdot\mathrm{e} &= h_{\mathbf{a}}(\mathbf{z}^* - \mathbf{t}_2) - \mathrm{p}\cdot\mathrm{e} && (5.1)\\ &= h_{\mathbf{a}}(\mathrm{e}^*\cdot\mathbf{s} + \mathbf{y} - \mathbf{t}_2) - \mathrm{p}\cdot\mathrm{e} && (5.2)\\ &= h_{\mathbf{a}}((\mathrm{e} - \mathrm{t}_1)\cdot\mathbf{s} + \mathbf{y} - \mathbf{t}_2) - \mathrm{p}\cdot\mathrm{e} && (5.3)\\ &= \mathrm{p}\cdot\mathrm{e} - \mathrm{p}\cdot\mathrm{t}_1 + \mathrm{x} - h_{\mathbf{a}}(\mathbf{t}_2) - \mathrm{p}\cdot\mathrm{e} && (5.4)\\ &= \mathrm{x} - \mathrm{p}\cdot\mathrm{t}_1 - h_{\mathbf{a}}(\mathbf{t}_2). && (5.5) \end{aligned}$$

Therefore, we have $\mathrm{H}(h_{\mathbf{a}}(\mathbf{z}) - \mathrm{p}\cdot\mathrm{e}, M) = e$ and $\mathrm{BS.Verif}(\mathrm{p}, M, (\mathbf{z}, \mathrm{e})) = 1$.

Moreover, each rejection sampling in step 2 succeeds after $G = \exp^{12/\delta + 1/(2\delta)^2}$ repetitions, with δ calculated from $\eta = \omega(T\sqrt{\log(n)})$, T is the norm of the vector e in step 2, we can rewrite $\eta = \delta T$ and then we have $\delta \geqslant k\sqrt{\log n}$.

□

The following lemma is a slightly modified version of the **Lemma** 2.

Lemma 14. *We have* $\Delta(\bar{D}_{\mathrm{R}^m,\mathrm{y},\sigma,B}, \bar{D}_{\mathrm{R}^m,0,\sigma,B}) \leqslant \frac{1}{1-2^{-\Omega(n)}} \cdot \frac{1}{2^{\omega(\log n)}}$.

Proof. This lemma argue about the statistical distribution of two gaussian distribution, but in our case to enforce the perfect correctness, we have to deal with truncated gaussian. The statistical distance between the truncated gaussian is

$$\frac{1}{2}\sum_{\mathbf{x}\in\mathrm{R}^m} |\bar{\rho}_{\beta,\mathbf{z}^*,D}(\mathbf{x}) - \bar{\rho}_{\beta,0,D}(\mathbf{x})| = \frac{1}{2}\sum_{\|\mathbf{x}\|\leqslant D} |\bar{\rho}_{\beta,\mathbf{z}^*}(\mathbf{x}) - \bar{\rho}_{\beta,0}(\mathbf{x})|$$

since if $\|\mathbf{x}\| > D, \bar{\rho}_{\beta,\mathbf{z}^*,D}(\mathbf{x}) = \bar{\rho}_{\beta,0,D}(\mathbf{x}) = 0$. By definition of $\bar{\rho}_{\beta,\mathbf{z}^*,D}$, we have:

$$\frac{1}{2}\sum_{\|\mathbf{x}\|\leqslant D} |\bar{\rho}_{\beta,\mathbf{z}^*,D}(\mathbf{x}) - \bar{\rho}_{\beta,0,D}(\mathbf{x})| = \frac{1}{2}\sum_{\|\mathbf{x}\|\leqslant D} \left|\frac{\rho_{\beta,\mathbf{z}^*}(\mathbf{x})}{\Phi_{\beta,\mathbf{z}^*}(D)} - \frac{\rho_{\beta,0}(\mathbf{x})}{\Phi_{\beta,0}(D)}\right|.$$

By **Lemma** 1, we have $\Phi_{\beta,0}(D) = Pr_{\mathbf{x}\leftarrow D^m_{\mathrm{R},\beta}}[\|\mathbf{x}\| \leqslant D] \geqslant 1 - 2^{-\Omega(n)}$, since $\sqrt{n}.\beta \leqslant D$. By the same argument, we have that $\|\mathbf{x}+\mathbf{z}^*\| \leqslant \|\mathbf{x}\|+\|\mathbf{z}^*\| \leqslant \sqrt{n}.(\beta+\sigma) \leqslant D$ for $\mathbf{x} \leftarrow D^m_{\mathrm{R},\beta}$, then we have $\Phi_{\beta,\mathbf{z}^*}(D) = Pr_{\mathbf{x}\leftarrow D^m_{\mathrm{R},\beta,\mathbf{z}^*}}[\|\mathbf{x}\| \leqslant D] \geqslant 1 - 2^{-\Omega(n)}$.

Finally, we have:

$$\begin{aligned}
\frac{1}{2}\sum_{\mathbf{x}\in\mathrm{R}^m} |\bar{\rho}_{\beta,\mathbf{z}^*,D}(\mathbf{x}) - \bar{\rho}_{\beta,0,D}(\mathbf{x})| &\leqslant \frac{1}{1-2^{-\Omega(n)}}.\frac{1}{2}\sum_{\|\mathbf{x}\|\leqslant D} |\rho_{\beta,\mathbf{z}^*}(\mathbf{x}) - \rho_{\beta,0}(\mathbf{x})| \\
&\leqslant \frac{1}{1-2^{-\Omega(n)}}.\frac{1}{2}\sum_{\mathbf{x}\in\mathrm{R}^m} |\rho_{\beta,\mathbf{z}^*}(\mathbf{x}) - \rho_{\beta,0}(\mathbf{x})| \\
&\leqslant \frac{1}{1-2^{-\Omega(n)}}.\frac{\|\mathbf{z}^*\|}{\beta} \\
&\leqslant \frac{1}{1-2^{-\Omega(n)}}.\frac{1}{2^{\omega(\log n)}}
\end{aligned}$$

□

We give now the blindness proof, the property is ensured thanks to the rejection sampling technique as well as the above lemma.

Theorem 6 (Blindness)**.** BS *is statistically blind.*

Proof. As per experiment $\mathrm{Exp}^{\mathrm{blind}}_{S^*,\mathrm{BS}}$, the adversarial signer outputs two messages M_0, M_1 and interacts with two users $\mathcal{U}(\mathrm{p}, M_b)$, $\mathcal{U}(\mathrm{p}, M_{1-b})$ after a secret coin flip $b \leftarrow \{0,1\}$. We show that these users do not leak any information about their respective messages.

To prove that the adversary has no advantage to associate the signatures with the messages, we prove that the distributions of the corresponding transcripts and signature can not be linked to the messages to be signed by the attacker. During an iteration of the blind signature issuing protocol, the transcript obtained by the malicious signer is a commitment x, a blind challenge e^* and a signature $\mathbf{z}^*$ delivered to the user. The outputs of the user are the message M, the challenge e and the blind signature $\mathbf{z}$.

What we have to do is to analyze the distributions of each one of these elements and prove that the two transcripts $\mathrm{x}_b, \mathrm{e}^*_b, \mathbf{z}^*_b$ are independent of the signatures $M_b, \mathrm{e}_b, \mathbf{z}_b$, for $b \in \{0,1\}$.

— **The commitment** x_b: The commitment is generated at the beginning of the protocol, using a gaussian distribution centered on 0, and of variance γ depending only on the security parameter n. It is then easy to see that these elements do not give any advantage to the attacker.

— **The blinded challenge** e_b^*: we have $e_b^* = e_b - t_1^b$, here the element e_b is part of the signature, then the distribution of e_b^* must be independent of the distribution of e_b. To ensure this property, we use the **Theorem** 1 on the rejection sampling. It means that after applying the rejection sampling on the element e_b^*, its distribution becomes independent of the element e_b, and then gives us the desired property.

— **The blinded signature** $\mathbf{z}_b^*$: The last element which composes the transcript is the signature computed by the signer $\mathbf{z}_b^* = \mathbf{z}_b + \mathbf{t}_2^b$, depending on the actual blind signature output by the user. In order to ensure the blindness of this element, we use the **Lemma** 14. This lemma states that the statistical distance between the blind signature $\mathbf{z}_b$ and the polynomial vector $\mathbf{t}_2^b$ is less than $\frac{1}{1-2^{-\Omega(n)}} \cdot \frac{1}{2^{\omega(\log n)}}$, which is negligible. Since the vector $\mathbf{t}_2^b$ is generated independently of the other elements of the protocol, then it concludes on the blindness property of our protocol.

We have then proved that all the elements composing the transcripts follow distributions that are indeed independent of any signature issued by a user at the end of a blind signature generation. We can conclude that our scheme is statistically blind since all the arguments used in the proof are statistical. □

What remains to prove is the one-more unforgeability security of the blind signature scheme, that is only conjectured in our case. This property asks that a user, interacting l times with an authority, is not able to output $l + 1$ blind signature. Considering the complicated history of this property on lattices, we start by providing a complete analysis of the issue emerging in the constructions preceding the scheme of [Hau+20], we then develop our own proof with a complete description of its mechanisms.

5.3 One-more unforgeability proof

For a long time the unforgeability property considered for blind signatures was the existential unforgeability. However Pointcheval and Stern [PS00] pointed out that this security notion was far to be exhaustive for the case of blind signatures, since it does not take into account the fact that the attacker is allowed to interact several times with the signer before to output its forgery. This is why a new notion, called one-more unforgeability, has been introduced by Pointcheval and Stern [PS00] to encompass the different scenarios that can be encountered in the real-world use-cases of blind signature schemes.

5.3.1 Definition and first proof

The one-more unforgeability property has been introduced in 1996 with an article from Pointcheval and stern [PS96a], they described then the first provably secure blind signature scheme. This work combined to the Eurocrypt 1996 paper of the same authors [PS96b] (that were regrouped into [PS00]) was groundbreaking in the world of provable security for signature schemes, on top of blind signature schemes. Relying on the novel model of random oracle by Bellare and Rogaway [BR93], Pointcheval and Stern introduced a new tool, called forking lemma, that has been generalized by Bellare and Neven [BN06] to contexts other than signature schemes. This tool will be intensively used by almost all signature schemes in the random oracle model for the following years.

The main goal of the proof, in the context of blind signatures, is to design a simulator interacting with a successful one-more unforgeability attacker, and expecting thanks to him, to solve a hard problem (such as the discrete logarithm of the factorization problem in [PS00]). Their result needs to take as a basis a blind signature protocol that enjoys the property of witness indistinguishability. This property means that for a randomly chosen secret key associated to a given public key, it exists at least one other secret key associated in the same way to the same public key. Moreover using one or another secret key is indistinguishable for anyone that does not manipulate the secret key.

Concretely, thanks to this witness indistinguishability property, a simulator (impersonating a real signer) can run an attacker, acting as a user, into a black-box simulation and expect the attacker to output a one-more forgery. In order to not let the attacker notice that he corresponds with a simulator instead of a true signer, the simulation needs to be indistinguishable of an actual iteration of the signing protocol. Moreover, the reduction makes use of the so-called forking lemma.

Intuitively, this lemma applied on blind signatures argues that if an attacker succeeds in the one-more unforgeability game and outputs a "one-more" signature, he can then be "rewinded" to the point where this forgery has been done, and by giving him different oracle answers, expects that it output a second "one-more" signature different from the first one. Thanks to those two different forgeries, the simulator can compute, with a non-negligible probability, a solution to a hard problem (typically, the SIS problem on lattices, or the discrete logarithm) using the properties inherited from the zero-knowledge proofs. However to succeed the reduction, some variable χ, depending on the transcripts and signatures, must differ in each one of these forgeries, allowing the simulator to get a solution to a hard problem thanks to these χ variables.

The argument used by [PS00] to evaluate the probability of the above event on the variable χ, mainly relies on the lemmas 8 and 9 of [PS00]. These lemmas build a function ϕ that takes as input a valid transcript of an interaction between a simulator and an attacker, and outputs a different valid transcript corresponding to another interaction with the same attacker and simulator, but which is indistinguishable to the first transcript from the attacker's point of view, thanks to the witness indistinguishable property of the protocol.

Finally thanks to this ϕ function and by using well-chosen sets of events, it is possible to evaluate the probability that the attacker outputs two one-more forgeries, such that the knowledge of these forgeries allows the simulator to finalize the reduction with a solution to the given hard problem. While this last part is generic and does not create difficulties in the lattice setting, the ϕ function detailed before is a lot more problematic and need some adaptations.

Coming to the topic of lattice-based blind signature, all existing schemes (as well as our) make use of this framework of proof using the forking lemma to achieve the one-more unforgeability property.

5.3.2 Problems in early constructions and solutions

The problem is that, Lemmas 8 and 9 developed in [PS00] are not immediately transposable to lattice-based blind signatures, as pointed by [Hau+20]. The reason is firstly linked to the fact that the elements are not chosen in the same set. Indeed the elements of the Okamoto-Schnorr scheme are integers chosen in a finite group $\mathbb{Z}_q$, while in most lattice-based schemes, elements are polynomial, or vector of polynomials chosen in the polynomial ring $\mathrm{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$. Then as the [PS00] ϕ function has been designed for the integer ring, it first has to be transposed to the case of polynomials.

But then arise the second and main problem of such a transposition into the lattice settings. Concerning the Okamoto-Schnorr protocol, a secret key can be any element in the finite group $\mathbb{Z}_q$, while in the lattices, a secret key is, most of the time, restricted to be a short element $\mathbf{s}$ of the ring R_q^m. The problem here is that, in the proof process, we have to define and use a function ϕ which sends a valid secret key $\mathbf{s}$ into an element $\mathbf{s}'$ which also has to be short to be a valid secret key. The initial [PS00] ϕ function has not been designed for that.

In order to overcome this problem, the authors of [Hau+20] defined the torsion free element $\mathbf{z}^*$, which is a non zero element of the kernel of the public key $\mathbf{a}$. Given this ele-

ment, the image by ϕ of the secret key $\mathbf{s}$ is $\phi(\mathbf{s}) = \mathbf{s}' = \mathbf{s}+\mathbf{z}^*$. They then use a sort of noise flooding to ensure that the new secret key $\mathbf{s}'$ remains sufficiently short to be a valid secret key. However the secret key $\mathbf{s}$ is not the only element modified by the function ϕ. As in [PS00] the commitment $\mathbf{y}$ also has an image by ϕ defined as $\phi(\mathbf{y}) = \mathbf{y}' = \mathbf{y} - \mathbf{z}^* \cdot \mathrm{e}$, so that they have to use the same noise flooding argument to ensure that the new element $\mathbf{y}'$ is short enough. Indeed this sort of noise flooding is the main reason of the inefficiency of the resulting scheme in [Hau+20], while they achieve a valid one-more unforgeability proof, they need a global enlargement of their elements to be sure with overwhelming probability that the image of the function ϕ still corresponds to a valid execution of the protocol.

Concerning our own function ϕ, instead of considering a generic non zero element $\mathbf{z}^*$ for any secret key $\mathbf{s}$, we link the secret key $\mathbf{s}$ to the function ϕ. Indeed for a given secret key $\mathbf{s}$, by the witness indistinguishability of the protocol, we know that it exists at least one other element $\mathbf{s}'$ in the set of the secret key, such that $\mathrm{p} = h_{\mathbf{a}}(\mathbf{s}) = h_{\mathbf{a}}(\mathbf{s}')$. Then we can consider that $\mathbf{z}^* = \mathbf{s}' - \mathbf{s}$, meaning that we ensure that the image $\mathbf{s}' = \mathbf{s} + \mathbf{z}^*$ of the secret key $\mathbf{s}$ is still a valid secret key. We then denote ϕ as $\phi_{\mathbf{s}}$ since the function ϕ differs with the secret key $\mathbf{s}$.

For the commitment $\mathbf{y}$, we use the same transformation as the function ϕ from [Hau+20] by sending $\mathbf{y}$ to $\mathbf{y} - \mathbf{z}^* \cdot \mathrm{e}$, but in our case we do not have the problem with the norm of the image as in [Hau+20]. In fact the uniform distribution does not allow the elements to exceed the boundary of the set on which the elements are picked. However in a gaussian distribution we do not have this bound restriction, since every element in the ring R_q^m has a non zero probability to be picked. It means that regardless what value is taken by $\mathbf{y} - \mathbf{z}^* \cdot \mathrm{e}$ it remains a possible candidate for the commitment generated by the signer in the first move of the protocol.

However, even if we succeed to define a function ϕ that sends a valid transcript to another valid transcript, the gaussian distribution induces a problem in the proof. Indeed, in the proof of [PS00] appears an inequality between two probabilities that is correct considering an uniform distribution, but cannot be proved in the same way using a gaussian one. The reason is that the probability inequality comes from a comparison in the number of elements, that compose the events under which the probabilities are evaluated. Indeed this size comparison can be transformed into a probability inequality only if the elements are uniformly distributed. The gaussian distribution does not profit this property and then the proof can not be achieved in the same way. This is why our one-more unforge-

ability security rely on a conjecture that this probability inequality remains true even for gaussian distributions.

Using this function $\phi_{\mathbf{s}}$ we overcome the problem solved by the noise flooding technique in the scheme of [Hau+20], while achieving a much better efficiency, since the elements of our scheme do not have to be oversized. The details of this part of the proof are given later in Definition 18 and Lemma 18. In our one-more unforgeability proof, the end of the proof of [PS00] can be developed in the same way to get a valid proof for this security property.

5.3.3 Our proof

One-more Unforgeability. The reduction in our scheme is done on the k-SIS problem, unlike to others lattice-based blind signature schemes, which are based on the SIS problem. In fact the addition of the trapdoor in our scheme removes the ability of the simulator to sign the requests sent from the attacker during the one-more unforgeability experiment as it is done in previous schemes. We then need to give him some SIS solutions to perform some rejection sampling in the signing step. We expect at the end of the simulation that the simulator is able to build a new SIS solution, linearly independent of those he gets during the simulation.

To simulate blind signatures queries, we first need the help of a k-SIS oracle. Indeed, this one provides us k short vectors in the kernel of a matrix $\mathbf{a}$ which we use to simulate the signatures in relation with the rejection sampling, we claim that the environment of the simulator is well simulated, since the probability that the simulator can answer all the signing queries is $(1-(1-\frac{1}{G_2})^k)^{q_{sign}}$ which holds with overwhelming probability if $q_{sign} = O(\exp k)$, meaning that $q_{sign} = O(\exp n)$ since we can take $k = O(n)$. We also require that it exists at least two secret keys corresponding to a given public key p_k.

Lemma 15 (Adapted from [Rüc10, lemma 3.6])**.** *Let $h \in \mathcal{H}(\mathrm{R}, m)$. For every secret key $\mathbf{s} \leftarrow_{\$} \mathrm{R}_3^m$, there is a second $\mathbf{s}' \in \mathrm{R}_3^m \backslash \{\mathbf{s}\}$ with $h(\mathbf{s}') = h(\mathbf{s})$ (with overwhelming probability).*

We fit in the proof [Rüc10, lemma 3.6] by replacing d_s by 1.

In fact, the goal is to assume that the attacker $\mathcal{A}$ will provide a one-more signature on a secret key $\mathbf{s}'$ different from the real one $\mathbf{s}$, used by our simulator. Moreover, to hide the secret key the simulator is using, we need a witness indistinguishable signature protocol.

Lemma 16 (Adapted from [Rüc10, lemma 3.7])**.** *Let $h \in \mathcal{H}(\mathrm{R}, m)$ and $\mathrm{p} \in \mathrm{R}$. For any message M and any two secret keys $\mathbf{s}, \mathbf{s}' \in \mathrm{R}_3^m$ with $h(\mathbf{s}) = \mathrm{p} = h(\mathbf{s}')$, the resulting protocol views $(\mathrm{x}_1, \mathrm{e}_1^*, \mathbf{z}_1^*)$ and $(\mathrm{x}_2, \mathrm{e}_2^*, \mathbf{z}_2^*)$ are indistinguishable.*

It means that the malicious verifier cannot distinguish whether the prover uses one of at least two possible secret keys $\mathbf{s}, \mathbf{s}' \in h^{-1}(\mathrm{p}) \cap \mathrm{R}_3^m$.

We expect that the attacker forges at least one signature that does not correspond to a signer's transcript. We then apply the Forking Lemma to extract knowledge about the secret key corresponding to the one-more forgery. The reduction uses this knowledge to solve the k-SIS problem, we show that the solution built by the k-SIS adversary is independent of the k vectors given by the k-SIS oracle with probability $\frac{\dim(ker(\mathbf{a}))-k}{\dim(ker(\mathbf{a}))} \in O(1)$.

Since the function family $\mathcal{H}(\mathrm{R}, m)$ compresses the domain R_3^m, we have all the secret keys which collide with at least one other secret key.

We finally apply the Forking Lemma to extract a "one-more" solution of the k-SIS problem $h_{\mathbf{a}}(\mathbf{v}) = 0$.

As mentioned before, we need the following conjecture to reach the one-more unforgeability security.

Conjecture 1. *Let the uplet $(\mathbf{s}, \mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_l\}, \rho, \mathbf{h})$ denotes a successful interactions between an attacker on the one-more unforgeability and a simulator impersonating a signer, with $\mathbf{s} \in \mathrm{R}_3^m$ the secret key, $\{\mathbf{y}_1, \ldots, \mathbf{y}_l\} \in D^l_{\mathrm{R}_q^m, \sigma}$ the commitments, ρ the attacker random tape and $\mathbf{h}$ the random oracle answers. We then have:*

$$\Pr[\exists i \in [l+1] \text{: } \mathbf{z}_i - \mathbf{s} \cdot \mathrm{e}_i \neq \mathbf{c}_i] \geqslant \Pr[\forall i \in [l+1] \text{: } \mathbf{z}_i - \mathbf{s} \cdot \mathrm{e}_i = \mathbf{c}_i \cap \exists i \in [l+1] \text{:} \mathbf{z}_i - \mathbf{s}' \cdot \mathrm{e}_i \neq \mathbf{c}_i]$$

With $\mathbf{z}_i$ the signature corresponding to the interaction $(\mathbf{s}, \mathbf{y}_i, \rho, \mathbf{h})$, e_i the challenge of the i-th interaction, $\mathbf{s}' \in \mathrm{R}_3^m$ verifying $h_{\mathbf{a}}(\mathbf{s}) = h_{\mathbf{a}}(\mathbf{s}')$ and $\mathbf{c}_i = \arg\max_{\mathbf{c}} \Pr_{\mathbf{h}'}[(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') \cap \mathrm{e}_i = \mathrm{h}'_j \cap \mathbf{z}_i - \mathbf{s} \cdot \mathrm{e}_i = \mathbf{c} | (\mathrm{h}_1, \ldots, \mathrm{h}_{j-1}) = (\mathrm{h}'_1, \ldots, \mathrm{h}'_{j-1})]$.

Intuitively the inequality in the conjecture means that, the probability that it exists at least one signature among the $l+1$ output by the adversary, such that the elements $\mathbf{z}_i - \mathbf{s} \cdot \mathrm{e}_i$ does not reach its most probable value, is bigger than the probability that these same values $\mathbf{z}_i - \mathbf{s} \cdot \mathrm{e}_i$ always reached their most probable value while, when applying the function ϕ the uplet $(\mathbf{s}', \mathcal{Y}' = \{\mathbf{y}'_1, \ldots, \mathbf{y}'_l\}, \rho, \mathbf{h})$ lies in the previous case.

Theorem 7 (One-more unforgeability)**.** *Let Sig be the signature oracle. Let T_{Sig} and T_H be the cost functions for simulating the oracles Sig and H.* BS *is $(t, q_{sign}, q_H, \delta)$-one-more unforgeable if Ring k-$SIS_{q,m,D}$ is (t', δ')-hard with $t' = 2t$ and non-negligible δ' if δ is non-negligible.*

Proof. Towards contradiction, we assume that there exists a successful forger $\mathcal{A}$ against one-more unforgeability of BS with non-negligible probability δ. Using $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ solving the k-SIS problem on R_q.

The idea is that the forger will forge a one-more signature twice, considering that in the second forgery it uses the same random tape for the forger $\mathcal{A}$ but different answers to the oracle queries compare to the first one. These hypothesis are essentials for the success of the attack, since we assume that the new one-more forgery is done on the same oracle query as in the first forgery, but the answers to these same queries are different, so we can build, upon these different answers on a same query, the "one-more" solution.

Reduction.

Setup. $\mathcal{B}$ gets a matrix $\mathbf{a}$ and k short vectors $\mathbf{v}_1, \dots \mathbf{v}_k$ of its kernel from a k-SIS oracle. $\mathcal{B}$ stores the values $\mathbf{v}_1, \dots \mathbf{v}_k$ in a list $L_{\mathbf{v}}$ and initializes a list $L_{\mathrm{H}} \leftarrow \emptyset$ of query-hash pairs in $(\mathrm{R}_q \times \{0,1\}^*, \mathrm{R}_2)$. It chooses a secret key $\mathbf{s} \leftarrow_{\$} \mathrm{R}_3^m$ and sets $\mathrm{p} \leftarrow h_{\mathbf{a}}(\mathbf{s})$. Furthermore, it randomly pre-selects random oracle answers $\mathbf{h} = \{\mathrm{h}_1, \dots, \mathrm{h}_{q_H}\} \leftarrow_{\$} \mathrm{R}_2$ and a random tape ρ. It runs $\mathcal{A}(\mathrm{p}, \rho)$ in a black-box simulation.

Random Oracle Queries. On input (u, C), $\mathcal{B}$ looks up (u, C) in L_{H}. If it finds corresponding hash value e then it returns e. Otherwise, $\mathcal{B}$ selects the first unused e from the list $\mathrm{h}_1, \dots, \mathrm{h}_{q_H}$, stores $((\mathrm{u}, C),\mathrm{e})$ in L_{H}, and returns e.

Blind Signatures Queries. $\mathcal{B}$ acts according to a modified version of the BS protocol, it begins by sending a commitment $\mathrm{x} = h_{\mathbf{a}}(\mathbf{y})$ with a fresh element $\mathbf{y} \hookleftarrow D_\gamma$ for each signing request. When $\mathcal{B}$ receives a blind challenge e^*, it looks up for the first element $\mathbf{v}_1$ in the list $L_{\mathbf{v}}$ and compute $\mathbf{z}^* = \mathrm{e}^* \cdot \mathbf{s} + \mathbf{y} + \mathbf{v_1}$, then he performs the rejection sampling test, if the vector $\mathbf{z}^*$ does not pass this test, $\mathcal{B}$ restarts this step with the next vector $\mathbf{v} \in L_{\mathbf{v}}$, and so on until a rejection sampling test passes, then he stops and outputs the corresponding signature $\mathbf{z}^*$. When the attacker performs a new blind signature query, we assume that the previous signature has been generated using a vector $\mathbf{v}_i \in L_{\mathbf{v}}$, then for this new signature query, the simulator starts from the next vector $\mathbf{v}_{i+1} \in L_{\mathbf{v}}$, and restarts from $\mathbf{v}_1$ when

he has reached the end of the list $L_{\mathbf{v}}$. We avoid then to use the same vectors for each signature queries, since in practice the rejection sampling passes with high probability using one of the first vectors $\mathbf{v}_i \in L_{\mathbf{v}}$. The probability that $\mathcal{B}$ is able to output a valid signature is $1-(1-1/G_2)^k$.

Output. Eventually, $\mathcal{A}$ stops and outputs $(M_1,(\mathbf{z}_1,\mathrm{e}_1)),\ \ldots\ ,(M_m,(\mathbf{z}_m,\mathrm{e}_m))$, $l+1=m$ with $l=q_{sign}$, for distinct messages.
Then the simulator $\mathcal{B}$ guesses the index of the one-more signature $f \leftarrow_{\$} [m]$ such that $\mathrm{h}_J = \mathrm{e}_f$ for some $J \in [q_H]$, we will denote (u_f, C_f) the corresponding query. Then, $\mathcal{B}$ starts over, running $\mathcal{A}(\mathrm{p},\rho)$ with random oracle answers $\mathbf{h}' = \mathrm{h}_1,\ldots,\mathrm{h}_{J-1},\mathrm{h}'_J,\ldots,\mathrm{h}'_{q_H}$ for a fresh set $\mathrm{h}'_J,\ldots,\mathrm{h}'_{q_H} \leftarrow_{\$} \mathrm{R}_2$. Both $\mathcal{A}$ and $\mathcal{B}$ are run with the same random tape as in the first run. Among other values, $\mathcal{A}$ outputs $(M'_f,(\mathbf{z}'_f,\mathrm{e}'_f))$, with (u'_f,C'_f) the oracle query for the answer e'_f, and $\mathcal{B}$ returns $(\mathbf{z}_f-\mathrm{e}_f\cdot\mathbf{s})-(\mathbf{z}'_f-\mathrm{e}'_f\cdot\mathbf{s})$ if $(\mathrm{u}'_f,C'_f)=(\mathrm{u}_f,C_f)$ (or equivalently $J'=J$ with $\mathrm{h}_J=\mathrm{e}_f$ and $\mathrm{h}_{J'}=\mathrm{e}_{f'}$) in an attempt to solve k-SIS on R_q. If $(\mathrm{u}'_f,C'_f)\neq(\mathrm{u}_f,C_f)$ (or $J\neq J'$), the reduction returns $\perp$.

Analysis.

The environment of the attacker is statistically indistinguishable from an actual iteration of the signing protocol, indeed the signatures sent by the simulator $\mathcal{B}$ are not generated exactly as in the signing protocol, but in an indistinguishable way. They are computed as $\mathbf{z}^* = \mathrm{e}^*\cdot\mathbf{s}+\mathbf{y}+\mathbf{v}$, with $\mathbf{v}\in L_{\mathbf{v}}$, instead of as the output of the function PreSample, but using Theorem 1, the statistical distance between a signature generated by the simulator and a signature generated by a genuine signer is $\frac{2^{-100}}{G_2}$, with $G_2=O(1)$, then is negligible. Moreover at least one vector $\mathbf{v}\in L_{\mathbf{v}}$ allows the simulator to compute a signature, well-formed thanks to the rejection sampling, with probability $(1-(1-1/G_2)^k)^{q_{sign}}$, then let be Sim the event that the simulation is well performed, we have $\Pr[\mathrm{Sim}] = \frac{1-2^{-100}}{G_2}\cdot(1-(1-1/G_2)^k)^{q_{sign}}$.

Lemma 17. *If $k=n$, then $k<\dim(ker(\mathbf{a}))$ and $q_{sign}\leqslant O(\exp(k))=O(\exp(n))$, then we have that the event* Sim *holds with overwhelming probability.*

Proof. Let $c=1-\frac{1}{G_2}$, $k=n$ and

$$q_{sign}=\exp\left(\log\left(-\log\left(1-\frac{1}{poly\,(n)}\right)\right)-(\log(2)+k\log(c))\right)=O\left(\exp(k)\right),$$

then we have

$$q_{sign} \leqslant \frac{-\log(1-\frac{1}{poly(n)})}{2c^k} \leqslant \frac{-\log(1-\frac{1}{poly(n)})}{\frac{c^k}{1-c^k}} \leqslant \frac{-\log(1-\frac{1}{poly(n)})}{\log(\frac{1}{1-c^k})} \leqslant \frac{\log(1-\frac{1}{poly(n)})}{\log(1-c^k)}.$$

We then have $q_{sign} \leqslant \frac{\log(1-\frac{1}{poly(n)})}{\log(1-c^k)}$, then $q_{sign}(\log(1-c^k)) \geqslant \log(1-\frac{1}{poly(n)})$ and $(1-c^k)^{q_{sign}} \geqslant 1-\frac{1}{poly(n)}$. We can conclude that the event Sim holds with overwhelming probability. □

We assume that $\mathcal{A}$ breaks one-more unforgeability. So, at least one of the output signatures is not obtained via an interaction. The probability that $\mathcal{B}$ guesses the index f of this signature correctly is at least $1/(l+1)$. Since e_f is a random oracle answer, we have $\mathrm{e}_f = \mathrm{e}'_f$ with probability $1/2^n$, which is negligible.

Applying the forking lemma, we know that with probability $\delta_{frk} \geqslant \epsilon \cdot (\epsilon/q_H - 1/2^n)$ with $\epsilon = \delta - 1/2^n$, $\mathcal{A}$ is again successful in the one-more unforgeability experiment and outputs $(M'_f, (\mathbf{z}'_f, \mathrm{e}'_f))$ using the same random oracle query as in the first run. Therefore, we know that $(h_{\mathbf{a}}(\mathbf{z}_f - \mathrm{p} \cdot \mathrm{e}_f), M_f) = (h_{\mathbf{a}}(\mathbf{z}'_f - \mathrm{p} \cdot \mathrm{e}'_f), M'_f)$. Now, we turn to solving the ring k-SIS problem.

We have to show that $\mathbf{z}_f - \mathrm{e}_f \cdot \mathbf{s} - (\mathbf{z}'_f - \mathrm{e}'_f \cdot \mathbf{s}) \neq 0$ with $h_{\mathbf{a}}(\mathbf{z}_f - \mathrm{e}_f \cdot \mathbf{s} - (\mathbf{z}'_f - \mathrm{e}'_f \cdot \mathbf{s})) = 0$. The last equality is verified from the previous paragraph. Concerning the first inequality, it is important that the protocol is witness indistinguishable (**Lemma 3**), i.e., the adversary does not recognize whether the simulator used one of at least two possible $\mathbf{s}, \mathbf{s}'$ (**Lemma 2**). We next introduce some notations that will be helpful to explain the rest of the proof. We finally develop the proof in the same way as in the original one-more unforgeability proof of [PS00].

Notations.

For the rest of the proof, we define the following elements: let the uplet $(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})$ define a transcript between the simulator $\mathcal{B}$ and the attacker $\mathcal{A}$. It is composed of the secret key $\mathbf{s}$, the l commitments generated during the signature queries $\mathcal{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_l)$, the random tape of the attacker ρ and the random oracle answers $\mathbf{h} \in \mathrm{R}_2^{q_H}$ with $\mathbf{h}_j$ the limitation of $\mathbf{h}$ to his j first elements $\mathbf{h}_j = \{\mathrm{h}_1, \ldots, \mathrm{h}_j\}$, then a given uplet $(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})$ defines the whole interaction between the simulator and the attacker. We also define $\mathcal{W}$ as the set of successful uplet in the one-more unfirgeability game, meaning that if $(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{W}$ then the attacker involved in the corresponding transcript succeeds to

win the one-more unforgeability game, and for each uplet in the set $\mathcal{W}$, we define the variable $\chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})$ that we denote χ.

Finally we define a quantity which evaluates the probability for the rewinded forgery on the transcript $(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}')$, that its variable χ' takes a given value $\mathbf{c}$ and that all the indexes are well formed to permit the success of the reduction. This quantity is necessary to study the probability for the k-SIS attacker $\mathcal{B}$ to win his game. Let $\lambda_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{c}) = \Pr_{\mathbf{h}'}[(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') \in \mathcal{W} \cap (Ind_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = j) \cap \chi(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = \mathbf{c}]$ and $\mathbf{c}_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = \arg\max_{\mathbf{c}} \lambda_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{c})$, with $J = Ind_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})$ the index such that $\mathrm{e}_f = \mathrm{h}_J$. We also define the variable $C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = \mathbf{c}_{f, Ind_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})$. These variables are going to help us to study the probabilities of the two following subset of $\mathcal{W}$: the subset $\mathcal{G}$ whose elements satisfy $\chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})$ for all f and the subset $\mathcal{F}$ which is the complement of $\mathcal{G}$ in $\mathcal{W}$ that are used to argue about the probability to get a collision.

We next define a function ϕ in the same way as the [[PS00], Lemma 8 and Lemma 9], to argue that the variables $\chi = \chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = \mathbf{z}_f - \mathrm{e}_f \cdot \mathbf{s}$ and $\chi' = \chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = \mathbf{z}'_f - \mathrm{e}'_f \cdot \mathbf{s}$ will be sensitive to the modified random oracle answers for indexes $\geqslant J$.

New ϕ function.

We recall that the attacker $\mathcal{B}$ against the collision of the hash function $h_{\mathbf{a}}$ wins his game, if the attacker $\mathcal{A}$ against the one-more unforgeability of the blind signature succeeds to output two one-more forgeries such that the variables χ associated to these forgeries are different. In order to quantify the probability to have these variables χ taking different values, we make use of the two subsets $\mathcal{G}$ and $\mathcal{F}$ and especially how they behave compared to each other, then we define the following $\phi_{\mathbf{s}}$ function in the way we discussed, in the beginning of this section, with the interest to solve the security issues while achieving a better efficiency:

Definition 18. *We define the function $\phi_{\mathbf{s}}$, such that for a fixed $\mathbf{s} \in \mathrm{R}_3^m$ we know that there exists $\mathbf{s}' \in \mathrm{R}_3^m$, with $\mathrm{p} = h_{\mathbf{a}}(\mathbf{s}) = h_{\mathbf{a}}(\mathbf{s}')$ by the witness indistinghuishability of the protocol. Let $\mathbf{t} = \mathbf{s}' - \mathbf{s}$, then the function $\phi_{\mathbf{s}}$ is*

$$\begin{aligned} \phi_{\mathbf{s}} \colon (\mathrm{R}_3^m, (\mathrm{R}_q^m)^l, \Omega, (\mathrm{R}_2)^{q_H}) &\to (\mathrm{R}_3^m, (\mathrm{R}_q^m)^l, \Omega, (\mathrm{R}_2)^{q_H}) \\ (\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) &\mapsto (\mathbf{s}', \mathcal{Y}', \rho, \mathbf{h}) \end{aligned}$$

Where $\mathcal{Y}' = \mathcal{Y} - \mathbf{e} \cdot \mathbf{t} = (\mathbf{y}_1 - \mathrm{e}_1 \cdot \mathbf{t}, \ldots, \mathbf{y}_l - \mathrm{e}_l \cdot \mathbf{t})$, with $\mathbf{e} = (\mathrm{e}_1, \ldots, \mathrm{e}_l)$ the set of challenges generated during the signature queries.

Lemma 18. *The function $\phi_{\mathbf{s}}$ fixes the whole interaction between the simulator and the attacker, furthermore we have that for $(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{W}$, then $\phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{W}$. Moreover, this function is injective on the set $\mathcal{W}$.*

Proof. We have to verify that the values seen by the attacker are the same when using the tuple $(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})$ or its image by $\phi_{\mathbf{s}}$.

1. **The public key**: the public key verifies $\mathrm{p} = h_{\mathbf{a}}(\mathbf{s}) = h_{\mathbf{a}}(\mathbf{s}) + h_{\mathbf{a}}(\mathbf{t}) = h_{\mathbf{a}}(\mathbf{s}')$, since $h_{\mathbf{a}}(\mathbf{t}) = 0$. Then for each tuple, the attacker is provided with the same public key.
2. **The commitment**: During the first step of the blind signature protocol, the signer generates a commitment and sends the hash of this commitment. Concretely, each commitment $\mathbf{y}$ is sent on $\mathbf{y}' = \mathbf{y} - \mathrm{e} \cdot \mathbf{t}$ by $\phi_{\mathbf{s}}$, we then have that $h_{\mathbf{a}}(\mathbf{y}') = h_{\mathbf{a}}(\mathbf{y}) - \mathrm{e} \cdot h_{\mathbf{a}}(\mathbf{t}) = h_{\mathbf{a}}(\mathbf{y})$. Then the attacker does not notice the change of value on $\mathbf{y}$.
3. **The signature**: The signatures $\mathbf{z}^*$ are sent to $\mathrm{e} \cdot \mathbf{s}' + \mathbf{y}' = \mathrm{e} \cdot (\mathbf{s} + \mathbf{t}) + \mathbf{y} - \mathrm{e} \cdot \mathbf{t} = \mathrm{e} \cdot \mathbf{s} + \mathbf{y} = \mathbf{z}^*$. Then the value of the signature $\mathbf{z}^*$ are the same for each execution.

The function $\phi_{\mathbf{s}}$ is injective. By contradiction suppose that it exists $(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \neq (\mathbf{s}', \mathcal{Y}', \rho', \mathbf{h}')$ such that $\phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = \phi_{\mathbf{s}}(\mathbf{s}', \mathcal{Y}', \rho', \mathbf{h}')$. This implies that $(\mathbf{s} + \mathbf{t}, \mathcal{Y} - \mathbf{e} \cdot \mathbf{t}, \rho, \mathbf{h}) = (\mathbf{s}' + \mathbf{t}, \mathcal{Y}' - \mathbf{e}' \cdot \mathbf{t}, \rho', \mathbf{h}')$. We directly have that $\rho = \rho'$ and $\mathbf{h} = \mathbf{h}'$, concerning the secret keys, we have $\mathbf{s} + \mathbf{t} = \mathbf{s}' + \mathbf{t}$ then $\mathbf{s} = \mathbf{s}'$ since $\mathbf{t} \neq 0$. Moreover with the above proof, we know that the function ϕ fix the same execution then we have that $\mathbf{e} = \mathbf{e}'$ and $\mathcal{Y} = \mathcal{Y}'$, we then get a contradiction. □

Our function $\phi_{\mathbf{s}}$ is very similar to the one from [Hau+20]. However we avoid the need of the huge growth on the elements, induced by the fact that due to the imperfect correctness, a valid secret key can be sent on an invalid one, and the same problem occurs for the commitment.

We conclude here on the probability of the simulator to break the ring k-SIS problem on the polynomial vector $\mathbf{a}$.

Now we develop and use the following lemma that has been introduced in [PS00], in order to study the relation between the $\phi_{\mathbf{s}}$ and the sets $\mathcal{G}$ and $\mathcal{F}$, which helps us to evaluate how the variable χ behaves.

Lemma 19. *For fixed $(\mathbf{s}, \mathcal{Y}, \rho)$ we have the probability:*

$$\Pr_{\mathbf{h}}[(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{G} \ \cap \ \phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{G}] \leqslant \frac{q_H^{l+1}}{q}.$$

Proof. To prove the above theorem, suppose that it exists a uplet $(\mathbf{s}, \mathcal{Y}, \rho)$ such that the probability:

$$\Pr_{\mathbf{h}}[(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{G} \ \cap \ \phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{G}] > \frac{q_H^{l+1}}{q}.$$

Then there exists a $l+1$ uplet $(J_1, \ldots, J_{l+1}) \in \{1, \ldots, q_H\}^{l+1}$ and a l uplet $\tilde{e}_1, \ldots, \tilde{e}_l \in \mathrm{R}_2^l$ such that $\forall f \in [l+1]$ and $\tilde{f} \in [l]$:

$$\Pr_{\mathbf{h}}[(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{G} \ \cap \ \phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{G} \cap Ind_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = J_f \cap (e_{\tilde{f}} = \tilde{e}_{\tilde{f}})] > \frac{1}{q^{l+1}}.$$

From the above probability, it must exist two distinct oracles $\mathbf{h}, \mathbf{h}' \in \mathrm{R}_2^{l+1}$ that verifies the condition. Moreover since every term, besides those indexed by $J_f, f \in [l+1]$, is fixed. It means that there is at least one index $J_f \in [l+1]$ such that $\mathrm{h}_{J_f} \neq \mathrm{h}'_{J_f}$, we denote j the smallest of such indexes, then, $\mathbf{h}_{j-1} = \mathbf{h}'_{j-1}$ and $\mathrm{h}_j \neq \mathrm{h}'_j$. Then we have that $C_i(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = C_i(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}')$ and:

$$\begin{aligned}
C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) &= \mathbf{z}_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) - \mathbf{s} \cdot \mathrm{h}_j \\
&= \mathbf{z}_f(\phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})) - \mathbf{s} \cdot \mathrm{h}_j \\
&= \mathbf{z}_f(\phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})) - \mathbf{s} \cdot \mathrm{h}_j + \mathbf{t} \cdot \mathrm{h}_j - \mathbf{t} \cdot \mathrm{h}_j \\
&= \mathbf{z}_f(\phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})) - (\mathbf{s} + \mathbf{t}) \cdot \mathrm{h}_j + \mathbf{t} \cdot \mathrm{h}_j \\
&= C_f(\phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})) + \mathbf{t} \cdot \mathrm{h}_j.
\end{aligned}$$

Using the same arguments, we have:

$$C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = C_f(\phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}')) + \mathbf{t} \cdot \mathrm{h}'_j.$$

Finally we have:

$$C_f(\phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})) + \mathbf{t} \cdot \mathrm{h}_j = C_f(\phi_{\mathbf{s}}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}')) + \mathbf{t} \cdot \mathrm{h}'_j.$$

However since $C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}')$ and $\mathbf{t} \neq 0$, then we have $\mathrm{h}_j = \mathrm{h}'_j$ which contradicts the hypothesis $\mathrm{h}_j \neq \mathrm{h}'_j$. □

What remains to prove, is that the probability for a given $(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{W}$ to verify

$(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{F}$ is non negligible and to conclude on the probability to get a collision on $h_{\mathbf{a}}$.

We can decompose the set $\mathcal{G}$ into two subsets, firstly $\mathcal{G}^g$ composed of the elements of $\mathcal{G}$ mapped to $\mathcal{G}$ by $\phi_{\mathbf{s}}$ and secondly the subset $\mathcal{G}^f$ composed of the elements of $\mathcal{G}$ mapped to $\mathcal{F}$ by $\phi_{\mathbf{s}}$. We then have that $\Pr[\mathcal{G}] = \Pr[\mathcal{G}^g] + \Pr[\mathcal{G}^b]$, from Lemma 12, we have $\Pr[\mathcal{G}] \leqslant \frac{q_H^{l+1}}{q} + \Pr[\mathcal{G}^f] \leqslant \frac{q_H^{l+1}}{q} + \Pr[\mathcal{F}]$. The last inequality stands using the conjecture 1.

Using the fact that $\mathcal{W} = \mathcal{G} \cup \mathcal{F}$ with $\Pr[W] = \epsilon$, the above probability becomes:

$$\Pr[\mathcal{F}] \geqslant \frac{1}{2}(\epsilon - \frac{q_H^{l+1}}{q}).$$

To conclude on the evaluation of success, we define the two types of events:

$$\mathcal{F}_f = \{(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{F} \cap C_f \neq \chi_f\} \text{ and } \mathcal{F}_{f,j} = \{(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \in \mathcal{F}_f \cap Ind_f = j\}.$$

Then we have the inequality $\sum_{f=1}^{l+1} \Pr[\mathcal{F}_f] \geqslant \Pr[\cup_{f+1}^{l+1} \mathcal{F}_f] = \Pr[\mathcal{F}]$. It means that there exists at least one index $f \in [l+1]$ such that $\Pr[\mathcal{F}_f] \geqslant \Pr[\mathcal{F}]/(l+1)$. We now assume that we have chosen such and index f.

We define the set $J_f = \{j | \Pr[\mathcal{F}_{f,j} | \mathcal{F}_f] \geqslant \frac{1}{2q_H}\}$, we then have:

$$\Pr[\cup_{j \in J_f} \mathcal{F}_{f,j} | \mathcal{F}] = \Pr[Ind_f \in J_f | \mathcal{F}_f] \geqslant \frac{1}{2}.$$

The above inequality is true, since $\Pr[Ind_f \in J_f | \mathcal{F}_f] = \sum_{j \in J_f} \Pr[\mathcal{W}_j | \mathcal{W}]$ with $\mathcal{W}_j$ the event $\mathcal{W}$ with $Ind_f = j$. The above probability is equal to $1 - \sum_{j \notin J_f} \Pr[\mathcal{W}_j | \mathcal{W}]$, we have that the complement of J_f is composed of less than q_H elements, then $\Pr[\cup_{j \in J_f} \mathcal{F}_{f,j} | \mathcal{F}] = \Pr[Ind_f \in J_f | \mathcal{F}_f] \geqslant 1 - q_H \cdot \frac{1}{2q_H} \geqslant \frac{1}{2}$.

We also have the probability $\forall j \in J_f$, $\zeta_{i,j} = \Pr[\mathcal{F}_{f,j}] \geqslant \frac{1}{4q_H(l+1)}(\epsilon - \frac{q_H^{l+1}}{q})$.

Now that we have evaluated the probability for the variable χ to take a value different of the most likely one, we state a lemma equivalent to the the lemma 2 from [HKL19], which gives the probability to obtain two different values χ for the two runs of the attacker $\mathcal{A}$. We first recall the splitting lemma, which is used to prove the next lemma:

Lemma 20 (Splitting lemma from [PS00, Lemma 1])**.** *Let $A \subset X \times Y$ such that $\Pr[(x,y) \in A] \geqslant \epsilon$. For any $\alpha < \epsilon$, define*

$$B = \{(x,y) \in X \times Y | \Pr_{y' \in Y}[(x,y') \in A] \geqslant \epsilon - \alpha\} \text{ and } \bar{B} = (X \times Y) \backslash B,$$

then the following statements hold:

1. $\Pr[B] \geqslant \alpha$.
2. $\forall (x, y) \in B, \Pr_{y' \in Y}[(x, y') \in A] \geqslant \epsilon - \alpha$.
3. $\Pr[B|A] \geqslant \alpha/\epsilon$.

Lemma 21 (Adapted from [HKL19, Lemma 2])**.**

$$\Pr_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{h}'}[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) \neq \chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') = j] \geqslant \zeta_{f,j}(\frac{\zeta_{f,j}}{4} - \frac{1}{2^n}).$$

Proof. Our proof is directly adapted from the proof of [HKL19, Lemma 2], we rewrite it here. We define the probability

$$\alpha_{f,j}(\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{d}) = \Pr_{\mathbf{h}'_j}[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \neq d \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') = j] \geqslant \mu_{f,j}(\mathbf{s},\mathcal{Y},\rho,\mathbf{h})/2,$$

with

$$\mu_{f,j}(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = \Pr_{\mathbf{h}'}[(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) \in \mathcal{F}_{f,j} \cap \mathbf{h}'_j = \mathbf{h}_j].$$

Let $B(s)$ be equals to 1 if the statement s is true and 0 else.

$$\begin{aligned}
&\Pr_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{h}'}[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) \neq \chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') = j] \\
&= \sum_{\mathbf{d}} Pr_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{h}'}[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = \mathbf{d} \cap \chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \neq \mathbf{d} \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') = j] \\
&= \sum_{\mathbf{d}} \mathrm{E}_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h}}[B(\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = \mathbf{d} \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = j) \cdot \alpha_{f,j}(\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{d})] \\
&\geqslant \frac{1}{2} \sum_{\mathbf{d}} \mathrm{E}_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h}}[B(\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = \mathbf{d} \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = j) \cdot \mu_{f,j}(\mathbf{s},\mathcal{Y},\rho,\mathbf{h})] \\
&= \frac{1}{2} \sum_{\mathbf{d}} Pr_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{h}'}[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = \mathbf{d} \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = j \cap (\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) \in \mathcal{F}_{f,j} \cap \mathbf{h}'_j = \mathbf{h}_j] \\
&= \frac{1}{2} Pr_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{h}'}[Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = j \cap (\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) \in \mathcal{F}_{f,j} \cap \mathbf{h}'_j = \mathbf{h}_j] \\
&= \frac{1}{2} \cdot \mathrm{frk} \\
&\geqslant \frac{1}{2} \cdot \zeta_{f,j} \cdot (\frac{\zeta_{f,j}}{4} - \frac{1}{2^n}).
\end{aligned}$$

What remains to prove is that $\alpha_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}) \geqslant \mu_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})$. Let define

$$\gamma_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}) = \Pr_{\mathbf{h}'}[\chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = \mathbf{d} \cap (\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') \in \mathcal{F}_{f,j} \cap \mathbf{h} = \mathbf{h}'].$$

Then we have two cases to consider:

Case 1 : Assume that:

$$\gamma_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}) \geqslant \frac{1}{2}\mu_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}).$$

First we can assume $\mathbf{d} \neq C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h})$ since if not, then: $\gamma_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}) \leqslant \Pr_{\mathbf{h}'}[\chi f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) = C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \cap (\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') \in \mathcal{F}_{f,j} \cap \mathbf{h} = \mathbf{h}'] = 0$.

$$\begin{aligned}
\alpha_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}) &= \Pr_{\mathbf{h}'_j}[\chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') \neq d \cap Ind_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = j] \\
&\geqslant \Pr[\chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = C_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}) \cap Ind_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = j] \\
&\geqslant \Pr[\chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = \mathbf{d} \cap Ind_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = j].
\end{aligned}$$

Using that $(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') \in \mathcal{F}_{f,j}$ implies $Ind_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = j$, then we have:

$$\begin{aligned}
\Pr[\chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = \mathbf{d} \cap Ind_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = j] &\geqslant \Pr[\chi_f(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') = \mathbf{d} \cap (\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}') \in \mathcal{F}_{f,j}] \\
&\geqslant \gamma_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}) \\
&\geqslant \frac{1}{2}\mu_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}).
\end{aligned}$$

Case 2:

$$\gamma_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}) < \frac{1}{2}\mu_{f,j}(\mathbf{s}, \mathcal{Y}, \rho, \mathbf{h}, \mathbf{d}).$$

$$\begin{aligned}\alpha_{f,j}(\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{d}) &= \Pr_{\mathbf{h}'_j}[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \neq d \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') = j] \\ &\geqslant \Pr[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \neq \mathbf{d} \cap (\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \in \mathcal{F}_{f,j} \cap \mathbf{h}_j = \mathbf{h}'_j] \\ &= \Pr[(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \in \mathcal{F}_{f,j} \cap \mathbf{h}_j = \mathbf{h}'_j] \\ &- \Pr[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') = \mathbf{d} \cap (\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \in \mathcal{F}_{f,j} \cap \mathbf{h}_j = \mathbf{h}'_j] \\ &= \mu_{f,j}(\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{d}) - \gamma_{f,j}(\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{d}) \geqslant \mu_{f,j}(\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{d})/2.\end{aligned}$$

This concludes the proof.

□

From this lemma we can conclude on the probability to get a collision on the hash function $h_{\mathbf{a}}$. Indeed the attacker $\mathcal{B}$ on the collision problem of the hash family $\mathcal{H}(\mathrm{R}_q, m)$ wins if:

$$\begin{aligned}&\Pr_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{h}'}[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) \neq \chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \cap h_{\mathbf{a}}(\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h})) = h_{\mathbf{a}}(\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}'))] \\ &= \sum_{j\in[q_H]} \Pr_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{h}'}[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) \neq \chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') = j] \\ &\geqslant \frac{1}{l+1} \cdot \max_{f\in[l+1]} \sum_{j\in[q_H]} \Pr_{\mathbf{s},\mathcal{Y},\rho,\mathbf{h},\mathbf{h}'}[\chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) \neq \chi_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') \cap Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}) = Ind_f(\mathbf{s},\mathcal{Y},\rho,\mathbf{h}') = j] \\ &\geqslant \max_{f,j} \frac{\zeta_{i,j}}{l+1} \cdot (\frac{\zeta_{i,j}}{4} - \frac{1}{2^n}) \\ &\geqslant \frac{1}{4q_H(l+1)^2}(\epsilon - \frac{q_H^{l+1}}{q})(\frac{1}{16q_H(l+1)}(\epsilon - \frac{q_H^{l+1}}{q}) - \frac{1}{2^n}).\end{aligned}$$

Now from this collision we have, $\|(\mathrm{e}_f - \mathrm{e}'_f)\cdot(\mathbf{s}'-\mathbf{s})\| \leqslant 4n^2 < q/2$ because $\|\mathrm{e}_f - \mathrm{e}'_f\| \leqslant 2\sqrt{n}$ and $\|\mathbf{s}' - \mathbf{s}\| \leqslant 2\sqrt{n}$. Thus, $(\mathrm{e}_f - \mathrm{e}'_f) \cdot (\mathbf{s}' - \mathbf{s}) = 0$ over $\mathbb{Z}[X]/(X^n + 1)$, which is an integral domain. Since $\mathrm{e}_f - \mathrm{e}'_f \neq 0$, we have the contradiction $\mathbf{s}' = \mathbf{s}$ and then a "one-more" k-SIS solution $\mathbf{z}_f - \mathrm{e}_f \cdot \mathbf{s} - (\mathbf{z}'_f - \mathrm{e}'_f \cdot \mathbf{s})$, the probability this vector is linearly independent of the k given by the k-SIS oracle is $\frac{(m-1).n-k}{(m-1).n}$ since the k+1-th solution is built independently of the k given to the simulator. The success probability is at least $\delta' \geqslant (1/4)(\frac{(m-1).n-k}{(m-1).n})(1-(1-\frac{1}{G_2})^k)^{q_{sig}+1}\frac{1}{4q_H(l+1)^2}(\epsilon - \frac{q_H^{l+1}}{q})(\frac{1}{16q_H(l+1)}(\epsilon - \frac{q_H^{l+1}}{q}) - \frac{1}{2^n})$, which is non-negligible if $k = O(\log(q_{sig}))$ and if δ is non-negligible.

□

Corollary 1. BS *is one-more unforgeable if solving Ring k-SIS*$_{q,m,D}$ *is hard for parameters* $m = \Omega(\log n), D = \sqrt{n}\beta + n^2\alpha = \mathrm{O}(n^3\sqrt{n}\, 2^{\omega(\log n)})$ *and* $q = 4mn\sqrt{n}\log(n)\beta.\mathrm{prime} = \Theta(n^6\, 2^{\omega(\log n)})$ *in lattices that correspond to ideals in* R.

As already mentioned in the beginning of this chapter, there is some use cases where the classical blind signature is not enough, and we must add some properties, like a monetary value or a date of validity, to the blind signature to fit in the real-world constraints.

5.4 Partially blind variant

We develop in the following a partially blind variant, which has a better efficiency than the schemes using the generic transformation from Abe and Okamoto [AO00] depicted in figure 5.9. This construction is very similar to the Okamoto-Schnorr blind signature (Figure 5.4) using the same parameters with an additional hash function $\mathcal{F}$ modeled as a random oracle. However, we can see that the elements generated in the first signer step and first user step has been doubled as well as the elements sent in the first and last exchange. In our construction there is no more elements generated and/or exchanged compared to our blind signature scheme.

The main difference between a partially blind signature and a basic blind signature is that the former necessitates to manage a common information info that is also signed by the signer during the protocol. In our scheme, we also make use of an additional hash function $\mathcal{F}$:$\{0,1\}^* \rightarrow \mathrm{R}_q$, seen as a random oracle, such that the hash value $\mathcal{F}(\mathsf{info})$ is integrated in the signature process. As usually done, we consider in our protocol that the signer and the user agreed on the common information before playing the protocol to generate the blind signature.

The key generation of the protocol is the same as in the blind version described in the previous section. The main difference is on the signing protocol. The common information is included by the signer on the generated signature $\mathbf{z}^*$, thanks to a lattice trapdoor that permits to include it on a short solution of an ISIS problem. The user can easily verify that the resulting signature really embed the right information.

5.4.1 Our Construction

The *KeyGen* part is the same as in the Blind Signature.

$\mathcal{S}(x, g, \mathsf{info})$		$\mathcal{U}(y = g^x, \mathsf{info}, M)$
$u, s, d \leftarrow_{\$} \mathbb{Z}_q$		
$z \leftarrow \mathcal{F}(\mathsf{info})$		
$a \leftarrow g^u, b \leftarrow g^s z^d \mod p$	$\xrightarrow{a, b}$	$t_1, t_2, t_3, t_4 \leftarrow_{\$} \mathbb{Z}_q$
		$z \leftarrow \mathcal{F}(\mathsf{info})$
		$a' = a g^{t_1} y^{t_2}$
		$b' = b g^{t_3} z^{t_4}$
		$\epsilon = \mathrm{H}(a' \|\| b' \|\| z, M)$
	$\xleftarrow{e}$	$e \leftarrow \epsilon - t_2 - t_4 \mod q$
$c \leftarrow e - d \mod q$		
$r \leftarrow u - cx \mod q$	$\xrightarrow{r, s, c, d}$	
		$\rho \leftarrow r + t_1 \mod q$
		$\omega \leftarrow c + t_2 \mod q$
		$\sigma \leftarrow s + t_3 \mod q$
		$\delta \leftarrow d + t_4 \mod q$
		If $\omega + \delta = \mathrm{H}(g^\rho y^\omega \|\| g^\sigma z^\delta \|\| z, M)$
		Output $(\rho, \omega, \sigma, \delta)$

Figure 5.9 – Abe Okamoto partially blind signature scheme [AO00]

Setup. The parameters also remain the same. The only modification is the addition of the hash function $\mathcal{F}$:$\{0,1\}^* \rightarrow$ R.
We recall that the algorithm PreSample allows to find short solutions of an ISIS problem, thanks to the trapdoor $\mathbf{T_a}$, i.e., writing $\mathbf{u} \leftarrow \text{PreSample}(\mathbf{T_a}, \mathcal{F}(\mathsf{info}), \sigma)$ means that $h_{\mathbf{a}}(\mathbf{u}) = \mathcal{F}(\mathsf{info}) \bmod q$ and that $\mathbf{u}$ is following a Gaussian distribution of parameter σ.

Signing. The signing protocol is shown in Figure 5.10.

Signer(s,$\mathbf{T_a}$)		**User(**p,M**)**
$\mathbf{y} \hookleftarrow D_\sigma^m$		
$\mathrm{x} \leftarrow h_{\mathbf{a}}(\mathbf{y})$	$\xrightarrow{\mathrm{x}}$	$\mathrm{t}_1 \hookleftarrow D_\alpha^n$
		$\mathbf{t}_2 \hookleftarrow D_{\mathrm{R},\beta}^m$
		If $\|\mathbf{t}_2\| > t\sqrt{n \cdot m} \cdot \beta$ then
		generate a fresh $\mathbf{t}_2$
		$\mathrm{e} \leftarrow \mathrm{H}(\mathrm{x} - \mathrm{p} \cdot \mathrm{t}_1 - h_{\mathbf{a}}(\mathbf{t}_2), M)$
		$\mathrm{e}^* \leftarrow \mathrm{e} - \mathrm{t}_1$
		Accept e^* with probability
		$\min(\frac{D_{\mathrm{R},\alpha}}{G_1 \cdot D_{\mathrm{R},\alpha,\mathrm{e}}}, 1)$
		Otherwise start over with a
$\mathbf{z}^* \leftarrow \text{PreSample}(\mathbf{T_a}, \mathrm{e}^* \cdot \mathrm{p} + \mathrm{x} + \mathcal{F}(\mathsf{info}), \sigma)$	$\xleftarrow{\mathrm{e}^*}$	fresh $\mathrm{t}_1 \hookleftarrow D_\alpha^n$
If $\|\mathbf{z}^*\| > t\sqrt{n \cdot m} \cdot \sigma$ then		
start over with a fresh		
$\mathbf{z}^* \leftarrow \text{PreSample}(\mathbf{T_a}, \mathrm{e}^* \cdot \mathrm{p} + \mathrm{x} + \mathcal{F}(\mathsf{info}), \sigma)$	$\xrightarrow{\mathbf{z}^*}$	$\mathbf{z} \leftarrow \mathbf{z}^* - \mathbf{t}_2$
		Output $(M, (\mathbf{z}, \mathrm{e}))$

Figure 5.10 – PBS protocol

Verification. The verification procedure BS.Verif$(\mathrm{p}, M, (\mathbf{z}, \mathrm{e}), \mathsf{info})$ outputs 1 iff $\|\mathbf{z}\| \leqslant D$ and $\mathrm{H}(h_{\mathbf{a}}(\mathbf{z}) - \mathrm{e} \cdot \mathrm{p} - \mathcal{F}(\mathsf{info}), M) = \mathrm{e}$.

We investigate now the security properties of our partially blind signature.

5.4.2 Security

The security of our partially blind variant can quite easily be adapted from the one of our "simple" blind signature scheme. We need to show that the common information does not harm the security of the protocol.

Completeness. The completeness of the protocol remains the same as the blind variant, except that the common information has been added. But this obviously does not change the completeness, then we have the following theorem:

Theorem 8 (Completeness)**.** *The scheme* PBS *is perfectly complete.*

Partial Blindness. The partial blindness is very similar to the blindness of the blind variant, but we need to make sure that the common information does not enable the signer to link a blind signature to its transcript.

In the partial blindness experiment, two blind signatures are generated on the same common information. In our case it exactly gives the same proof as the blindness security since the user part remains exactly the same. The only change in the protocol is the addition of a pre-image in the computation of the signature by the signer, but this new element does not change the actual distribution of the signature output by the signer, then it gives him no more advantage. As this is quite obvious, we do not repeat again the whole proof. We obtain the following theorem.

Theorem 9 (Blindness)**.** PBS *is statistically blind.*

One-more Unforgeability. Regarding the one-more unforgeability, the adaptation of the proof from our basic blind signature is not as straightforward. In fact we need to be sure that the user cannot output a $l+1$-th partially blind signature after having obtained l valid signatures. There are two cases to consider with the common information. In the first case, the malicious user outputs a partially blind signature with a common information which has not been output in any previous signature, and in the second case, the common information linked to the signature has already been output.

Considering the two cases, we can use the proof built for our blind signature, since all the elements linked to the common information info vanish when we construct the Ring k-SIS solution and then the proof remains valid. The main problem to solve is in the simulation, since the simulator has not the trapdoor but needs to compute a pre-image sampling of the element $\mathbf{u} = \mathcal{F}(\mathsf{info})$. In fact, this problem can be solved by programming the answer of the hash function $\mathcal{F}$. More precisely, we generate a small value $\mathbf{u} \hookleftarrow D_\sigma^m$

and program $\mathcal{F}(\mathsf{info}) = h_{\mathbf{a}}(\mathbf{u})$, then store the value $\mathbf{u}$ and use it again for each request of partially blind signature with info as common information.

Applying these modifications we can now easily transpose the proof of the one-more unforgeability of the blind signature to the partially blind variant.

Theorem 10 (One-more unforgeability)**.** *Let Sig be the signature oracle. Let T_{Sig}, T_H and $T_{\mathcal{F}}$ be the cost functions for simulating the oracles Sig, H and $\mathcal{F}$.* PBS *is $(t, q_{sign}, q_H, q_{\mathcal{F}}, \delta)$-one-more unforgeable if Ring k-SIS$_{q,m,D}$ is $(t', \delta'/2)$-hard with $t' = t+(q_H+q_{\mathcal{F}})^{q_{Sign}}(q_{Sign}T_{Sign}+ q_H T_H + q_{\mathcal{F}} T_{\mathcal{F}})$ and non-negligible δ' if δ is non-negligible.*

Chapter 6

Practical Post-Quantum Resistant E-Voting Scheme

The notion of on-line voting is appealing since the emergence of remote communications. However until now, there is no e-voting protocol that offers all the properties (security, efficiency...) satisfactory for such a sensitive topic. Still there exists some interesting constructions that have been used in real-world elections such as Votopia [CGT06] or Helios [Adi08] which was trialed during student elections, for example in Princeton and the Catholic University of Louvain. The International Association of Cryptographic Researcher (IACR) also adopted Helios to elect its Board.

In this chapter we investigate the construction of a post-quantum e-voting system built from a framework introduced by Fujioka et al. in 1992 [FOO92], which mainly relies on the well-known cryptographic primitive called a blind signature scheme. This framework contrasts from the current trend that makes use of homomorphic encryption, or mix-net system [CRS05; CMM19] to improve the efficiency of the tallying phase in addition to offer strong verifiability and privacy properties thanks to zero-knowledge proofs. However in post-quantum setting, a lack of efficiency of some of the primitives used in the two frameworks cited above leads us to investigate on new options for a practical e-voting protocol.

It is in this context that we propose our new e-voting protocol. We start by introducing the definition and properties expected from an e-voting protocol. We then describe the primitives that we use to build our protocol, they include the blind signature scheme developed in the previous chapter, a public key encryption scheme, and a threshold transformation to the two aforementioned cryptographic primitives. Finally we give a complete description of our e-voting protocol alongside with the security proofs.

6.1 Definition and constructions

We start by giving a definition of such a protocol and describing the security properties that an e-voting system should satisfy along with a discussion on what we expect for our own construction. We then give a quick overview of the existing post-quantum e-voting protocols and we finally introduce the framework on which we base our work

6.1.1 Generic definition of an e-voting scheme

We present now a definition of en e-voting protocol, including a description of the different parts involved in such a scheme and the different algorithms composing the protocol along with their behaviors.

An e-voting protocol needs several entities to work, we describe them in the following list:

— First we get a set of N eligible voters $\mathcal{V}_i$ for $i \in [N]$.

— We also need a set of p authorities $\mathcal{A}_j$ for $j \in [p]$, that will share the private election keys.

— Finally we need a bulletin board BB, that will collect the (valid) ballots cast by the voters. At the end of the election, the valid ballots will be tallied.

We take as a basis the definition of an e-voting protocol of Cortier et al. [Cor+14]. However, we will voluntarily omit some of the algorithms they introduce and will modify some of them to comply with our e-voting framework, which is interactive between the voter and the authorities:

Setup$(1^n, 1^p, 1^N) \to (\mathbf{pk}, \{[\mathbf{sk}]^i\}_{j\in[p]})$**:** The setup algorithm, takes as input the security parameter n with the number of authorities p and the number of voter N. It generates a couple of public/secret key $(\mathbf{pk}, \mathbf{sk})$, such that the secret key is generated and output in a threshold manner, where each authority gets a share $[\mathbf{sk}]^j, j \in [p]$.

Vote$(\mathcal{V}_i(v, \mathbf{pk}), \{\mathcal{A}_j([\mathbf{sk}]^j)\}_{j\in \mathrm{T}}) \to b$**:** In the voting phase, each voter $\mathcal{V}_i, i \in [N]$ wants to cast his voting choice v. This step includes an online phase between the voter and some set of $\mathrm{T} \subset \{1, \ldots, N\}$ authorities $\mathcal{A}_j$ (at least a threshold t of them) equipped with a share $[\mathbf{sk}]^j$ of the secret key. The ballot b is then output and cast anonymously in the bulletin board BB.

Validate(b, pk) $\rightarrow \{0,1\}$: On input a ballot b, anyone can check its validity with the help of the public key **pk**, this algorithm outputs 1 if the ballot is valid and 0 otherwise.

Box(BB, b) $\rightarrow BB$: It takes as input the current state of the bulletin board BB, along with a ballot b. It first checks the validity of b by performing the algorithm described above: **Validate**(b, **pk**). It updates BB by adding b if Validate outputs 1 and remains unchanged if it outputs 0.

Tally($BB, \mathbf{pk}, \{\mathcal{A}^j([\mathbf{sk}]^j)\}_{j\in \mathrm{T}}) \rightarrow r$: During the tallying phase a tallying authority opens all the ballots thanks to the recovery of the secret key **sk** from a set of T authorities. Finally it counts the valid votes and publishes the outcome r of the election.

Verify($BB, \mathbf{pk}, \mathbf{sk}, r) \rightarrow \{0,1\}$: The verify algorithm can be run by anyone to check the validity of the tally result r, it outputs 1 is the tallying is correct and 0 otherwise.

In our definition of e-voting protocol, we omit the credential phase, that gives the right to eligible voter to vote. Since we do not focus our work on this part, we consider that our e-voting protocol can be equipped with a strong authentication mechanism that allows every eligible voter to take part to the election, while protecting their integrity. We also do not include an algorithm allowing each voter to check that his vote is included in the bulletin board BB, since this later is made public, then anyone can have a reading access on it.

6.1.2 Security properties of an e-voting scheme

Next, we discuss the security properties a secure e-voting protocol should fulfill. Obviously, we want that the anonymity of each voter is granted. Indeed, some flavor of anonymity has been considered in the past works, from the property of ballot privacy to the voter privacy, we develop our own anonymity property. In the same time, we would like that the scheme is verifiable, meaning that the result claimed by the voting authority reflects the votes cast by honest voters. Finally we include a correctness property, ensuring that for an honestly generated vote, the validation steps always outputs 1.

Attacker Model. Before formally presenting the different security properties, we discuss the model of the attacker. Indeed, we base our proofs on a game-based model, during

which the attacker has access to some oracles. We first present the oracles available to the attacker, before defining the security properties.

The attacker can make use of the oracles that we list below, along with their behavior.

— $\mathcal{O}$corruptU(id): it checks that id corresponds to an eligible voter who is allowed to take part to the election. If this is not the case it halts, else it gives the ability to the attacker to vote in place of voter id.

— $\mathcal{O}$vote(id, v): it first checks whether voter id already voted or not. If this is the case it halts else it returns $b = \mathbf{Vote}(\mathcal{V}_{id}(v, \mathbf{pk}), \{\mathcal{A}^j([\mathbf{sk}]^j)\}_{j\in \mathrm{T}})$ and adds BB:= $BB \cup b$.

Now, for the correctness and verifiability properties, we rely on the corresponding definitions developed by Cortier et al. [Cor+14]. Concerning the anonymity, we develop our own property that we call ballot anonymity, which is similar to the privacy notion introduced by Kremer and Ryan [KR05].

Correctness

Concerning the correctness, we fit in with the definition of [Cor+14]. The idea of this property is that a genuinely generated ballot is always accepted into the bulletin board, and for an election where all parts behaves honestly, the result of the tally always corresponds to the votes cast by the voters. It means that for $(\mathbf{pk}, \mathbf{sk}) = \mathbf{Setup}(1^n, 1^p, 1^N)$ we need to have the two statements true

1. $\mathbf{Validate}(b, \mathbf{pk}) = 1$, for all $b = \mathbf{Vote}(\mathcal{V}_i(v, \mathbf{pk}), \{\mathcal{A}_j([\mathbf{sk}]^j)\}_{j\in \mathrm{T}})$.
2. $\mathbf{Verify}(\mathrm{BB}, \mathbf{pk}, \mathbf{sk}, r) = 1$, for a bulletin board $BB = \{b_1, \cdots, b_k\}, k \leqslant N$ with each b_i verifying $\mathbf{Validate}(b_i, \mathbf{pk}) = 1$.

Verifiability

The verifiability property is a fundamental security property needed in e-voting schemes and it has been the subject of a several papers in the e-voting literature that have been summarized in [Cor+16]. However, as for the above correctness, we rely on [Cor+14] to define the verifiability property. Before to describe it, we specify that we only consider partial tallying e-voting protocol, it means that the tallying phase is not performed in a single computation, but each ballots is open separately, then the resulting tally is computed step by step. Verifiability asks that the tallying result is consistent with the votes cast by honest voters. More precisely the output of the algorithm **Tally** should actually count :

1. The votes cast by honest voters, who checked that their ballot appeared in the Bulletin Board (their votes are denoted v^E), the set of their votes is denoted Event.
2. A subset of the votes cast by honest voters who did not check that their ballots appeared in the Bulletin Board (their votes are denoted v^A).
3. The votes cast by corrupted users, which are gathered in the set denoted $\mathcal{CU}$. At most n_C such voters should have been tabulated, where n_C denotes the number of corrupted voters taking part to the election (their votes are denoted v^C).

We also denote the set of votes cast by honest voters by Hvote. The verifiability property is then translated into the corresponding experiment, where we consider classical election with the result r being the sum of the votes cast for each candidates:

$\mathrm{Exp}^{\mathrm{ver}}_{\mathcal{A}}(n)$

$(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathrm{Setup}(1^n, 1^p, 1^N)$

$(BB, r) \leftarrow \mathcal{A}^{*\mathcal{O}\mathrm{corruptU}(),\mathcal{O}\mathrm{Vote}()}(\mathbf{pk})$

If $\mathrm{Verify}(BB, \mathbf{pk}, \mathbf{sk}, r) = 0$, return 0

If $\exists\{v_1^E, \cdots, v_{n_E}^E\} =$ {Event},
$\{v_1^A, \cdots, v_{n_A}^A\} =$ {Hvote} - {Event},
$\{v_1^C, \cdots, v_{n_C}^C\}$ with $0 \leqslant n_C \leqslant |\mathcal{CU}|$

Such that $r = \sum_{i=1}^{n_E} v_i^E + \sum_{i=1}^{n_A} v_i^A + \sum_{i=1}^{n_C} v_i^C$ return 0 else return 1

Figure 6.1 – Verifiability experiment

The verifiability experiment considers an attacker, who first obtains the election public key **pk** generated during the setup phase. Then he can make request to the corrupt oracle, which on input an honest voter V_{id}, gives to the attacker the ability to cast a vote on behalf of the voter id, and add this id to the list of corrupted voters. He also gets access to a voting oracle, which on input a valid id and a voting choice v, casts a valid ballot b corresponding to the voting choice v for the identity id, only if this identity has not already cast a vote.

The bulletin board will then contain the honest ballots generated by the voting oracle, ballots cast by honest voters, and the ballots generated by corrupted users controlled by the attacker. The attacker wins this experiment, if its claimed result of the election is valid, and does not correspond to the sum of the votes cast by the set of the n_E honest

$\mathrm{Exp}^{\mathrm{bind}}_{\mathcal{A}}(n)$

$(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathrm{Setup}(1^n)$
$(v_0, v_1, id_0, id_1) \leftarrow \mathcal{A}^*(\mathsf{find}, \mathbf{pk}, \mathbf{sk})$
$c \leftarrow_{\$} \{0, 1\}$
$\mathrm{state}_{\mathsf{issue}} \leftarrow_{\$} \mathcal{A}^{*\langle\cdot, \mathcal{V}_{id_0}(\mathbf{pk}, v_c)\rangle, \langle\cdot, \mathcal{V}_{id_1}(\mathbf{pk}, v_{1-c})\rangle}(\mathsf{issue}, \mathsf{state}_{\mathsf{find}})$
Let b_c and b_{1-c} be the outputs of $\mathcal{V}_{id_0}(\mathbf{pk}, v_c)$ and
$\mathcal{V}_{id_1}(\mathbf{pk}, v_{1-c})$, respectively.
$d \leftarrow_{\$} \mathcal{A}^*(\mathsf{guess}, b_c, b_{1-c}, \mathsf{state}_{\mathsf{issue}})$
Return 1 iff $d = c$

Figure 6.2 – Ballot indistinguishability experiment

voters who checked that their ballots are included in the bulletin board, plus the votes of a subset of the n_A honest voters who did not checked whether their ballots appeared in the bulletin board plus at most n_C votes cast by corrupted voters.

Then a voting protocol is said to be verifiable if there is a negligible function $\mu(n)$ such that:

$$\mathrm{Succ}^{\mathrm{ver}}(\mathcal{A}^*) = \Pr[\mathrm{Exp}^{\mathrm{ver}}_{\mathcal{A}^*}(n) = 1] < \mu(n)$$

Ballot anonymity

Vote secrecy is another fundamental security property that any voting protocol should fulfill. It asks that the voting choice of a voter remains private during and after the end of the election. In this chapter we depart from the classical definition which is termed ballot privacy, that has been the subject of an intensive research (resumed in [Ber+15]). Indeed in our e-voting protocol, each ballot will be anonymous, that is, it does not identify the voter who casts it. This departs from a lot of other voting protocols, where each ballot is directly linked to the voter who casts it, leading to the fact that in the tally procedure each individual ballot could not be open (or decrypt) otherwise this would leak for whom a voter voted. Our case can be seen similar as the privacy property defined in [KR05].

We then consider a privacy property, where the attacker needs to link the identity of the voter to his ballot. We define it below:

In the experiment in figure 6.2, an attacker $\mathcal{A}^*$ simulates the elections authorities and interacts with two voters $\mathcal{V}_{id_0}$ and $\mathcal{V}_{id_1}$. The possible votes v_0 and v_1 are chosen by the

attacker in the mode find. Then it plays the role of the authorities, such that the two voters $\mathcal{V}_0$ and $\mathcal{V}_1$ generate their vote, with its help, in the mode issue. Finally after that each voter outputs their respective ballots b_c and b_{1-c} in an anonymous way, the attacker tries to guess the token c, i.e. he tries to link each ballot to each voter.

Then a voting protocol satisfies ballot anonymity if there is a negligible function $\mu(n)$ such that :

$$\mathrm{Succ}^{\mathrm{priv}}(\mathcal{A}) = \Pr[\mathrm{Exp}^{\mathrm{priv}}_{\mathcal{A}}(n) = 1] < \mu(n).$$

6.1.3 Post-quantum constructions

To the best of our knowledge, there exists only 2 post-quantum constructions, both built from lattice-based primitives. The first scheme is based on fully-homomorphic encryption by Chillotti et al [Chi+16] and appears in 2016. The second one is using zero-knowledge proofs on top of homomorphic commitments by del Pino et al. [Pin+17] the next year.

Concerning the scheme of Chillotti et al. [Chi+16], the key idea is that they get rid of the zero-knowledge proofs that are inefficient in lattice-based setting, as already discussed in the group signature chapter. Indeed, their work is inspired by the e-voting protocol of Helios [Adi08] even if, contrary to their scheme, needs the use of zero-knowledge proofs. These zero-knowledge proofs first allow the voters to prove that their ballots are correctly formed, but also permit the tally authority to prove that the result of the election is correct.

In a nutshell, [Chi+16] uses a fully-homomorphic encryption to replace the zero-knowledge proofs on the voter's side, while they use publicly verifiable ciphertext trapdoors to overcome the absence of zero-knowledge proof on the authority side. However using fully-homomorphic encryption makes the resulting voting scheme quite inefficient as pointed by del Pino et al. [Pin+17]. Moreover this problem of efficiency may explains why implementations for the [Chi+16] scheme are lacking, while the construction of [Pin+17] is accompanied by such an implementation which allows anyone to represent how it can behave for real-world elections.

Concerning the construction of del Pino et al. [Pin+17], the most important difference is that they make use of the zero-knowledge proof contrary to [Chi+16]. In fact, the study of lattice-based zero-knowledge proofs has been intensive in the past 5 years with several advances in particular regarding their efficiency. This allows them to rely on a construction

that makes a trade-off between efficiency and security. In short, their construction focuses on the Fiat-Shamir framework of [Lyu12], in order to prove the knowledge of the multiple of a short element instead of the element itself.

In addition to the zero-knowledge primitive, they use the commitment scheme of [Bau+16] that benefits of an additive homomorphic property, which is very appealing in the e-voting context. Finally, as said above, they provided an implementation of their voting scheme, that permits to analyze the efficiency of their construction in a real-world scenario. Indeed, generating and casting ballot is about 8.5 ms, and the time needed in the authority side as well as the verification step is 0.15 sec.

However, their implementation considers only two candidates, while if we want to add more candidates, the efficiency weakens. Indeed, for 2^k possible candidates, the number of proofs needed is multiplied by a factor k, then the efficiency decrease from a logarithmic factor in the number of candidates. Our own constructions does not suffer this drawback, since the efficiency does not depend of the number of candidates considered.

6.1.4 Framework of Fujioka et al. and adaptations

We base our construction on the framework of Fujioka et al. [FOO92]. In this framework, the anonymity is granted by a cryptographic primitive, called blind signature, while everyone can verify that the outcome of the election is correct, since all the elements that are necessary to the tally will be made public at the end of the election.

Therefore, the main tool of the [FOO92] framework is a blind signature scheme, since it allows to preserve the anonymity of each participant, a requirement that is mandatory for any election, while it forbids voters from voting twice. This blind signature primitive combined to a commitment scheme, that prevents any partial result to leak before the end of the election, are the two primitives that allow [FOO92] to build a secure e-voting scheme.

Concretely to generate his vote, any voter begins by computing a commitment of its voting choice, in order to conceal it from other voters until the end of the election. Then, he authenticates to the voting authority in order to obtain a blind signature on the commitment of his vote. Both the commitment and the (blind) signature constitute the voter's ballot which is then (anonymously) sent to the Bulletin Board. This later only stores signed ballots and discard invalid ones (i.e. either ballot that are not signed or ballot with an invalid signature).

At the end of the election, all the voters have to anonymously open their commitment

(that is to reveal their vote and the random value used to generate the commitment), so that anyone can see their voting choice. However thanks to the blindness property, the anonymity is preserved since no one will be able to link a signed ballot to the voter who requested the corresponding signature (and therefore no one will be able to link a voter to his vote). Finally anyone can tally the result of the election, by counting the votes and verifying the validity of the blind signatures associated to the opened commitment.

The FOO voting scheme [FOO92] suffers from several major drawbacks. The main one is that all voters have to participate to the ballot counting process, as they have to open their commitment at the end of the election. Their scheme is not "vote and go" and would be unsuitable for real-life elections. Worst, the private key of the blind signature scheme is held by a single authority who could easily stuff the Bulletin Board by generating as many blind signatures (meaning valid but illegitimate ballots) as he wishes.

In order to overcome these issues, we next introduce two functionalities, namely a (post-quantum) threshold public-key encryption scheme as well as a threshold variant of our blind signature scheme.

6.2 Our construction

6.2.1 Cryptographic primitives

In this context of electronic voting system, we need several cryptographic tools that are the building blocks of our voting scheme. In this section, we first briefly mention the main tool that we are using, which is the blind signature scheme developed in the previous chapter. Secondly we also present an encryption scheme called dual Regev encryption scheme, based on the Ring-LWE problem [Reg05], that we transform in a distributed variant where the private key is shared among several authorities. Finally we present a tool from Bendlin et al. [BKP13] that we make use for the threshold transformation we mentioned above. The two threshold transformations are used to avoid the problem of a single authority holding the election private keys and who could either stuff the Bulletin Board with (valid but illegitimate) ballots or learn partial results of the election.

Blind signature

We make use the blind signature we introduced in the chapter 5 to equip our e-voting protocol.

Encryption scheme

Now we introduce the dual-Regev encryption scheme first described in [GPV08], we describe it on a polynomial ring R_q defined above. Moreover, any public key encryption post-quantum secure would fit into our e-voting protocol, then any NIST submission could replace the following construction:

Construction We describe below the different algorithms of the encryption scheme :

PK.Setup. The setup algorithm chooses integers n, m, q and two real α, β such that the dual-Regev encryption scheme on the polynomial rings is deemed secure (see [LPR13]).

PK.KeyGen(1^n). It start by sampling $\mathbf{s} \hookleftarrow D_{\mathrm{R}^m,\alpha}$ as the secret key and $\mathbf{a} \leftarrow_{\$} \mathrm{R}_q^m$ uniformly at random as part of the public key. Finally it computes and reveals $u = \mathbf{a}^T\mathbf{s} \in \mathrm{R}_q^m$. The secret key is then the element $e_k = \mathbf{s} \in \mathrm{R}_q^m$ and the public key is composed of the pair $m_k = (\mathbf{a}, u) \in \mathrm{R}_q^m \times \mathrm{R}_q$, it outputs (e_k, m_k) as the secret and public key pair.

PK.Encrypt(m, m_k). Given a message m from the message space R_2, and a public key $m_k = (\mathbf{a}, u)$, it chooses a vector $v \in \mathrm{R}_q$ uniformly at random, and output the ciphertext $(\mathbf{b} = \mathbf{a}v + \mathbf{e}, c = u \cdot v + e' + \lfloor q/2 \rfloor m) \in \mathrm{R}_q^m \times \mathrm{R}_q$ where $\mathbf{e} \hookleftarrow D_{\mathrm{R}^m,\beta}$ and $e' \hookleftarrow D_{\mathrm{R},\beta}$. The ciphertext is then composed of the pair $(\mathbf{b}, c) \in \mathrm{R}_q^m \times \mathrm{R}_q$.

PK.Decrypt($(\mathbf{b}, c), e_k$). Given a ciphertext $(\mathbf{b}, c) \in \mathrm{R}_q^m \times \mathrm{R}_q$ and a private key $\mathbf{s} \in \mathrm{R}_q^m$, it computes $\mu = c - \mathbf{b}^T\mathbf{s} = \mathbf{e}^T \cdot \mathbf{s} + e' + \lfloor q/2 \rfloor m$. Finally to recover the message m, it suffices to look after each coordinate of μ, if the i-th coordinate is closer to 0 than to $\lfloor q/2 \rfloor$ then the i-th bit of m is equal to 0 and 1 otherwise.

Security proof. The dual Regev system is IND-CPA secure, assuming that Ring-$\mathrm{LWE}_{n,q,D_{\mathrm{R},\beta}}$ is hard, see [LPR13].

However, concerning our use case of a voting system, the IND-CPA security is not sufficient, since what we need is a non-malleability security property (at least NM-CPA). The reason is that we do not want that a voter replays a ballot (or a variant) of an other voter (i.e. duplicate the voting choice of this voter) without knowing what the actual voting choice is. Concretely it means that for an encrypted vote $c = \mathbf{Encrypt}(v)$,

which is published on a public board, a different voter must not be able to compute $c' = \mathbf{Encrypt}(v)$ with $c' \neq c$ without knowing v.

In order to extend the IND-CPA property into a NM-ATK property (where ATK can be any attack type, such as CPA, CCA1 or CCA2), we use the well-known Fujisaki-Okamoto transformation [FO99; HHK17] . This transformation starts from a scheme which is IND-CPA secure and transforms it into a IND-CCA2 secure scheme. We note that the IND-CCA2 security property is equivalent to the NM-CCA2 security property [Bel+98] which is enough for our purpose.

However, for our purpose we will modify this public key encryption lightly, in order to split the secret key between several entities, but the encryption and decryption protocols remain the same, so this modification does not harm the security of the former encryption scheme.

Threshold tools and variants

In the original version of our blind signature scheme, there is only one signer who could easily, in the context of e-voting, stuff the Bulletin Board by adding as many valid (but illegitimate) ballots as he wishes, we therefore transform it into a threshold one, using the generic transformation of a trapdoor based signature scheme with strong trapdoor of [MP12], into a threshold trapdoor based signature scheme by [BKP13]. We notice that our transformation is applied on a blind signature scheme and not on a signature scheme.

However, we point out that the signer part of the blind signature is composed of two steps, a first one which is a generation of a commitment, which can be transformed in a threshold manner using Shamir secret sharing, and a trusted setup to share a gaussian vector. Concerning the second step of the signer, it consists on a classic Fiat-Shamir with abort signature, which can easily be transformed into a threshold one by means homomorphic properties of the Shamir secret sharing. In case of abort, the signer performs a GPV-like signature which is a generic signature scheme and can be transformed using the generic transformation of [BKP13] into a threshold scheme.

We would like to emphasize that the construction of [BKP13] is built on the integer ring $\mathbb{Z}$, but our blind signature relies on polynomial ring $\mathrm{R}_q = \mathbb{Z}_q[\mathrm{X}]/f(\mathrm{X})$, with f a polynomial of degree n. However the strong trapdoor construction can be adapted to this ring setting [MP12], and the Shamir secret sharing [Sha79] still works on this type of ring. Then the whole construction of [BKP13] can be adapted to the polynomial ring setting.

Threshold Functionalities. We notice that we consider the threshold transformation of [BKP13], which makes use of a trusted setup. Indeed their variants that works without trusted setup needs the use of multilinear maps that are a sensitive topic [GGH13; Che+15; Cor+15]. In fact, in practice the settings without trusted setup are mandatory, however the concrete parameters of the e-voting protocol can be generated long time before the election day.

The proof of the various protocols from [BKP13] are realized in the UC model [Can01], so that we just have to plug the threshold functionalities into our blind signature scheme, to obtain, by composability, a secure threshold variant of our blind signature scheme. Below we describe the two mains protocols, which are the KeyGen and the SampleZ protocol. Moreover we choose to give an informal description of the functionalities involved for these two protocols. The full construction can be found in the paper of [BKP13].

We consider p authorities, such that a threshold of t authorities is mandatory to execute the various functions developed below. Let $\mathbf{a}' \in \mathrm{R}_q^{m-k}$ be a uniformly distributed vector of polynomial and $\mathbf{T} \in \mathrm{R}_q^{(m-k)\times k}$ be a Gaussian-distributed matrix. Let $\{[\mathbf{T}]^i\}_{i\in[p]}$ be the shares of the polynomial matrix $\mathbf{T}$. Let us denote by $\mathbf{a}_1 = \mathbf{a}'^T \cdot \mathbf{T} \mod q$ and $\mathbf{a} = [\mathbf{a}'|\mathbf{a}_1]$. In fact the key generation mechanism is the same as described in Lemma 4, but performed on a polynomial ring as done on our blind signature scheme in the chapter 5.

$\mathcal{F}_{\text{Blind}}$: This functionality takes as input shares of an arbitrary value x and output fresh shares $[x]^i$ of this same value.

$\mathcal{F}_{\text{SampZ}}$: This functionality takes as input dimensions $h \times d$ and a gaussian variance z. It outputs shares $[\mathbf{Z}]^i$ of a gaussian distributed matrix $\mathbf{Z} \leftarrow D_z^{h\times d}$.

Threshold KeyGen protocol :

The KeyGen protocol is realised in the $\mathcal{F}_{\text{Blind}}, \mathcal{F}_{\text{SampZ}}$ model.

On input the tuple $(\mathbf{a}', h^* \in \mathbb{R}_q, z \in \mathbb{Z})$, each party i does

1. Call $\mathcal{F}_{\text{SampZ}}((m-k)\times k, z)$, then receive $[\mathbf{T}]^i$.
2. Call $\mathcal{F}_{\text{Blind}}(-\mathbf{a}'^T[\mathbf{T}]^i)$, then receive $[\mathbf{a}_1]^i$.
3. Broadcast $[\mathbf{a}_1]^i$ and reconstruct $\mathbf{a}_1 = \mathbf{a}'^T \cdot \mathbf{T} \mod q$ from the other shares.
4. Output $\mathbf{a} = [\mathbf{a}'|h^* \cdot \mathbf{g} + \mathbf{a}_1]$ as the public key and $[\mathbf{T}]^i$ as the private key of the authority i.

$\mathcal{F}_{\text{Gadget}}$: This functionality takes as input a coset value $v \in \mathrm{R}_q$ and outputs shares $[\mathbf{u}]^i \in \mathrm{R}^k$ of a gaussian distributed polynomial vector such that $\mathbf{g}^T \cdot \mathbf{u} = v$.

$\mathcal{F}_{\text{Correct}}$: This functionality generates for each $j \in [k]$ and $v \in \mathrm{R}_q$ queues $Q_{j,v}$ of at least B values in each queue, that will allow the signer to perform at least B pre-image of each vector $v \in \mathrm{R}_q$. Each queue $Q_{j,v}$ is composed by using the gadget functionality developed above and the shares $[\mathbf{T}]^i$ of the trapdoor such that each authority gets a share of $\mathbf{y}_{j,v} = \begin{bmatrix} \mathbf{T} \\ \mathbf{I} \end{bmatrix} (\mathbf{e}_j \otimes \mathbf{z}_{j,v})$ for $\mathbf{z}_{j,v} \in \Lambda_v^{\perp}(\mathbf{g}^T)$, with $\mathbf{e}_j$ the vector composed of 0 elements except the j-th coordinate equals to 1.

Then, in the sampling algorithm of [MP12], when we have to correct a perturbation to get a correct sample for a given syndrom $v \in \mathrm{R}_q$, the authorities recover a value in the corresponding queue $Q_{j,v_1}, \cdots, Q_{j,v_n}$.

$\mathcal{F}_{\text{Perturb}}$: The perturb algorithm in the threshold setting, is the same as in the standard setting, but the perturbation vector is then shared between the p authorities using the functionality $\mathcal{F}_{\text{SampZ}}$. then it takes as input a dimension $h \times d$ and a gaussian parameter z, it outputs $[\mathbf{P}]^i$ with $\mathbf{P} \leftarrow D_z^{h \times d}$.

SampleZ protocol. Using the functionalities $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$ defined above that are the threshold counterparts of the steps composing the **Preimage sampling** protocol described in the section 3.3.3, the SampleZ protocol generates a presample in the same way as the standard algorithm developed in [MP12], but with the threshold variants of the subalgorithms perturb and correct.

Threshold variants. Using the tools described above, we can transform our blind signature primitive into a threshold variant. A similar modification will be applied on the public key encryption scheme, but which is light since we do not need for our voting scheme a full threshold transformation. Indeed, for our voting scheme we do not consider a threshold decryption mechanism, meaning that we just have to modify the PK.KeyGen algorithm accordingly to include a sharing mechanism of the decryption secret key.

We now develop the modifications that are brought to the two primitives of blind signature and public key encryption:

— **TBS.KeyGen**$(1^n, 1^p)$. The threshold variant of the blind signature key generation algorithm generates the same elements. Namely, a public polynomial vector $\mathbf{a}$ with a

trapdoor $\mathbf{T}$ using a trapdoor generation algorithm and a random polynomial vector $\mathbf{s} \in \mathrm{R}_3^m$ with its image by the hash function such that $\mathrm{p} = h_{\mathbf{a}}(\mathbf{s})$. All these elements are generated in a distributed way by using the Threshold Keygen protocol described above to generate the trapdoor along with the vector $\mathbf{a}$ used in the hash function. Concerning the secret vector $\mathbf{s}$, the algorithm $\mathcal{F}_{\mathrm{SampZ}}$ is executed by each authority, in order to obtain $[\mathbf{s}]^i, i \in [p]$, they each then have to broadcast their public part $\mathbf{a} \cdot [\mathbf{s}]^i, i \in [p]$ to recover and output the public key p. Finally the algorithm outputs the public key $p_k = (\mathrm{p}, \mathbf{a})$ and the private key share $[s_k]^i = ([\mathbf{T}]^i, [\mathbf{s}]^i), i \in [p]$ to each authority $\mathcal{A}^i, i \in [p]$.

— **TBS.Sign**$(\{(\mathcal{A}^i([s_k]^i))\}_{i \in T}, \mathcal{V}(p_k, M))$. Considering a set of T signing authorities $\mathcal{A}^i, i \in [T]$, the signature algorithm is the same as the one depicted in figure 5.8 from the user's side. Concerning the signer view, firstly the commitment $\mathbf{y}$ is generated in a distributed manner using the algorithm $\mathcal{F}_{\mathrm{SampZ}}$, such that the authorities get a share $[\mathbf{y}]^i$ and distributively output the corresponding element $\mathrm{x} = h_{\mathbf{a}}(\mathbf{y})$ in the same way as it was done in the TBS.KeyGen algorithm for the pair $(\mathbf{s}, \mathrm{p})$. Concerning the signing step, from the signer's view, the first attempt of signature, which is a Fiat-Shamir like signature [Lyu12], is performed between the authorities thanks to the homomorphic property of the Shamir secret sharing, while the GPV-like [GPV08] signature generation is performed in a threshold manner using the SampZ protocol. Finally, the algorithm outputs the signature $\sigma = (M, \mathrm{e}, \mathbf{z})$.

— **TPK.KeyGen**$(1^n, 1^p)$. It generates $\mathbf{s} \hookleftarrow D_{\mathrm{R}^m, \alpha}$ as the secret key in a distributed way using the $\mathcal{F}_{\mathrm{SampZ}}$ algorithm, such that each authority $\mathcal{A}^i, i \in [p]$ gets a share $[\mathbf{s}]^i$. Concerning the public key, it chooses $\mathbf{a} \leftarrow_{\$} \mathrm{R}_q^m$ uniformly at random. Finally each authority computes and reveals $[u]^i = \mathbf{a}^T[\mathbf{s}]^i \in \mathrm{R}_q^m, i \in [p]$ such that u can be recovered and output publicly. The secret keys are then the elements $[e_k]^i = [\mathbf{s}]^i \in \mathrm{R}_q^m, i \in [p]$ and the public key is composed of the pair $m_k = (\mathbf{a}, u) \in \mathrm{R}_q^m \times \mathrm{R}_q$, it outputs (e_k, m_k) as the secret and public key pair.

6.2.2 Our scheme

As explained above, we chose to modify the [FOO92] framework in order to let voters "Vote and go" (i.e. they will not have to do any further action at the end of the election once they have cast their ballot) and to prevent ballot stuffing by a malicious authority. We start by replacing the commitment scheme with a public key encryption scheme, so that

the voting choices are encrypted instead of being committed. At the end of the election, the decryption key will be disclosed so that anyone will be able to decrypt the ballots and compute the result of the election. Moreover, thanks to the indistinguishability property of the encryption scheme, the votes will remain hidden until the end of the election.

Another issue, more focused on the security aspect, is that the private key of the blind signature scheme is given to a single authority in [FOO92], the same problem would arise for the encryption scheme if we give the private decryption key to a single authority. It means that if the authority owning this private decryption key is corrupted, then he can get partial results by decrypting the ciphertexts before the end of the election, which is not desirable for most elections. Another concern is the fact that the private key of the blind signature scheme is owned by a single authority in [FOO92], who could generate as many ballots as he wants and stuff the Bulletin Board with them. An obvious solution would be to transform these two primitives into threshold variants, so that these private keys would be shared among several authorities and not a single one.

Concerning the encryption scheme, the transformation considered turns it into a slight version of a threshold encryption scheme. Indeed, we just need to avoid that the secret key is given to a single authority, then only the key generation mechanism is impacted, while the encryption and decryption remains unchanged. The idea is that at the end of the election, at least a threshold of T authorities publish their shares, so that anyone can reconstruct the whole private key and decrypt the ciphertexts of the valid ballots included in the bulletin board and finally compute the outcome of the election.

The second transformation is heavier, since we need to transform a whole blind signature protocol into a threshold one. We start by the result of Bendlin et al. [BKP13] that exhibits a generic transformation of a signature scheme making use of the trapdoors of [MP12] into a threshold variant. Since the security of this transformation is proven on the universally composable (UC) model [Can01], then by composability our threshold variant remains secure. We finally get that the two operations composing the signer part (the commitment and the signing step) are done in a threshold way by communicating with, at least, t signing authorities.

Considering these modifications of the [FOO92] framework, we next describe the complete e-voting protocol that we build from the cryptographic primitives introduced above. First, a setup phase generates the parameters of the protocol, including the private and public keys of the cryptographic schemes. Next the voting phase is composed of two steps, the voter first encrypts his voting option (using the public election key) and then interacts

with (at least) t voting authorities to obtain a blind signature on his ciphertext. His ballot b is composed of a ciphertext $\mathbf{c}$ of his voting choice v along with a (blind) signature σ on $\mathbf{c}$ such that $b = (\mathbf{c}, \sigma)$. The Bulletin Board only accepts a ballot if σ is a valid signature on $\mathbf{c}$ and discards it otherwise. Then, in the counting phase, the tallying authorities reveal their share of the private encryption key, so that anyone can recover the corresponding decryption key and decrypt the ballots (the ciphertexts $\mathbf{c}$) to compute the result of the election. Auditing the election is easy. For this purpose, an interested voter first have to check that all the ballots collected by the Bulletin Board are valid (i.e. that the signatures σ are valid),that the decryption key published by the talliers is correct (i.e. corresponds to the public election key). He then has to decrypt all the ballots using the decryption key and computes the result of the election just as the talliers did.

Construction

We present now our e-voting construction:

Setup$(1^n, 1^p, 1^N)$**.** The setup algorithm has to generate two pairs of secret/public keys, one pair for the encryption scheme and another one for the blind signature scheme. Moreover these keys have to be generated in a threshold manner, for a number p of authorities, with a threshold number of t. Let us denote by $(s_k, p_k) \leftarrow \mathbf{BS.KeyGen}(1^n)$ and $(e_k, m_k) \leftarrow \mathbf{PK.KeyGen}(1^n)$, and by $[s_k]^i$ (resp $[e_k]^i$) the shares of the private blind signature key (resp encryption key). Then the setup algorithms outputs $\mathbf{pk} = (p_k, m_k)$ and $\mathbf{sk} = ([s_k]^i, [e_k]^i)_{i \in [p]}$.

Vote$(\mathcal{V}_i(v, \mathbf{pk}), \mathcal{A}^j([s_k]^j)_{j \in \mathrm{T}})$**.** The voting phase is split in two steps. In the first step, the voter $\mathcal{V}_i$ encrypts his vote $v \in \{0,1\}^*$ in $\mathbf{c} = \mathbf{PK.encrypt}(v, m_k)$ in an offline phase. Then in an online phase, he is first authenticated (to see whether he is an eligible voter who has not yet requested a blind signature from the voting authorities). The protocol aborts if the authentication failed or if the voter already requested a blind signature. He then interacts with voting authorities $\mathcal{A}^j$ to get a blind signature $\sigma = \mathbf{BS.Sign}(\{\mathcal{A}^j([s_k]^j)\}_{j \in \mathrm{T}}, \mathcal{V}_i(p_k, \mathbf{c}))$, with T a set of authorities of size at least t. Finally the voter outputs $(\sigma, \mathbf{c})$ as his ballot and casts it, anonymously, into the bulletin board BB.

Validate(b**, pk).** On input a ballot $b = (\sigma, \mathbf{c})$, anyone can check its validity by performing the verification algorithm of the blind signature $\mathbf{BS.Verify}(p_k, \mathbf{c}, \sigma)$, it outputs 0 if

the blind signature verification fails and 1 otherwise.

Box(BB, b). It takes as input the current state of the bulletin board BB, along with a ballot b. It first checks the validity of b by performing the algorithm described above: Validate(b, **pk**). It updates $BB \leftarrow BB \cup \{b\}$ if Validate outputs 1 and remains unchanged if it outputs 0.

Tally$(BB, p_k, \mathcal{E}^j([e_k]^j)_{j\in[p]})$**.** At the end of the election, at least a threshold of t authorities $(\mathcal{E}^j)_{j\in[t]}$ holding the shares of the decryption key e_k reveal publicly their share $([e_k]^j)_{j\in t}$, such that anyone can rebuild the decryption key e_k. Then for each ballot $(\sigma, \mathbf{c}) \in BB$, anyone can decrypt $\mathbf{c}$ and retrieve the vote $v = \text{PK.decrypt}(\mathbf{c}, e_k)$ of each voter, after verifying that $\text{BS.Verify}(\sigma, p_k) = 1$. Then it can tally and outputs the result r, which corresponds to the outcome of the election $\mathbf{r} = \{v_i\}_{i\in k}$ with $k \leqslant N$ the number of voter that output a valid ballot.

Verify$(BB, \mathbf{r}, e_k)$**.** The algorithm verify is straightforward, since the decryption secret key e_k is public, then anyone can check the validity of the result by decrypting all the ciphertexts $\mathbf{c}$ contained in the ballots $(\sigma, \mathbf{c}) \in BB$, with a valid blind signature σ and tally them to compare to the result $\mathbf{r}$ announced by the tallying authorities.

6.2.3 Security of our scheme

Correctness

Theorem 11 (Correctness)**.** *Since* BS *and* PK *are correct, then our e-voting scheme is correct.*

Proof. According to our definition of correctness, our e-voting protocol is correct if a ballot generated by an honest voter is accepted with overwhelming probability by the Bulletin Board, and if the result of an honest election, where every party behaves honestly, corresponds to the votes casts by the voters.

Concerning the first condition, in our e-voting system a ballot is accepted by the bulletin board if the blind signature that forms the ballot is valid. Then since the blind signature scheme we consider satisfies the correctness requirement of a blind signature, meaning that the verification step always outputs 1 for a truly generated blind signature,

then we obtain that the first condition for the correctness of our e-voting protocol, is fulfilled.

The second condition asks that for an honest election (meaning that all parties involved in the voting protocol behave honestly), the outcome of the election announced by the authorities corresponds to the votes cast by eligible voters. It means that all the ballots included in the bulletin board at the end of the election open to votes that were cast by legitimate voters. Since the public key encryption we consider satisfies the correctness requirement of a public key encryption scheme, then the second condition if fulfilled either, because the decryption mechanism will recover the correct vote for each decryption.

Since our e-voting protocol satisfies both conditions it is therefore correct. □

Verifiability

Theorem 12 (Verifiability)**.** *By using a strong authentication scheme, and since* BS *is one-more unforgeable, then our e-voting scheme is verifiable.*

Sketch of proof. The attacker in the experiment in figure 6.1 wins if he can output a result that is valid and that does not correspond to the voting choices of honest voters that checked their ballots, plus the voting choices of a subset of honest voters that did not check their ballots, plus the voting choices of a subset of corrupted users.

At the end of the election the ballots stored in the bulletin board can no longer be modified even if we consider a malicious bulletin board. The reason is that the bulletin board is made public, then ballots can not be replaced nor new ones can be added without being noticed. Then to win the game, the attacker has to act before the end of the election. It means that he either (1) has to impersonate an honest voter or (2) casts more valid votes (let say $n_C + 1$) than the number (n_C) of corrupted users.

(1) is deemed infeasible since this would mean that he has successfully broken the strong authentication scheme used by voters to authenticate themselves to the Voting Authorities.

(2) is not possible either since this would mean that the attacker could generate more valid blind signatures than requested (in other words, this would imply that he can break the one-more unforgeability security of our threshold blind signature scheme) or that there exists more than dishonest voting authorities (which could generate as many valid-but illegitimate-ballots as they wish).

Furthermore, he can not cheat after the end of the election, since the ability to tally

the election is made public, then anyone can tally on his own and check the claimed result; our voting protocol therefore satisfies the Verifiability requirement.

We further notice that, independently of the attacker behavior, the tally can not give two different results for two iterations of the Tally algorithm. Indeed a valid ballot casts by a honest voters could be opened on two different results or the blind signature could be valid on the first iteration and invalid in the second one. However this scenario can not happen since the decryption mechanism and the blind signature verification algorithm are both deterministic.

Ballot anonymity

Theorem 13 (Ballot anonymity)**.** *If* BS *is blind, then our voting scheme satisfies ballot anonymity requirement.*

Sketch of proof. According to the experiment 6.2, an attacker $\mathcal{A}^*$ on the ballot anonymity interacts with two valid voters with identities id_0 and id_1. He first selects two voting choices v_0 and v_1 and sends them to the two considered voters $\mathcal{V}_{id_0}$ and $\mathcal{V}_{id_1}$. In the mode issue, he interacts with these two voters, to help them generate their corresponding ballots. Indeed, on a coin flip $c \leftarrow_{\$} \in \{0,1\}$, the voters $\mathcal{V}_{id_0}$ and $\mathcal{V}_{id_1}$ outputs the ballots $(\sigma_{id_c}, \mathbf{c}_{id_c})$ and $(\sigma_{id_{1-c}}, \mathbf{c}_{id_{1-c}})$ respectively such that $\mathbf{c}_{id_c} = \text{PKE.encrypt}(v_c, m_k)$ and $\mathbf{c}_{id_{1-c}} = \text{PKE.encrypt}(v_{1-c}, m_k)$. Finally $\mathcal{A}^*$ tries to guess the bit c in the guess mode.

We then have to prove that the attacker $\mathcal{A}^*$ has a negligible advantage to win the above game. According to the protocol, each ballot does not include any information about the identity of each voter, since the encryption and blind signature schemes are performed only on the voting choices v_0 and v_1 that are strictly independent of the identities id_0 and id_1. It means that the only way to the attacker to win the game is to distinguish the ballots during the interactive part of the voting algorithm performed by the voters and the attacker $\mathcal{A}^*$. In fact this interactive part consists in the execution of the blind signature protocol, therefore if the attacker succeeds in linking a ballot b_c to the corresponding execution of the blind signature protocol, then this would mean that he can break the blindness requirement of the blind signature scheme. But since the blind signature scheme is *blind* with overwhelming probability, then our e-voting scheme satisfies the ballot anonymity requirement with the same probability.

We finally notice that, independently of the security properties considered above, the encryption scheme prevents any partial result to leak. Indeed the secret key to decrypt

the votes included in the ballots is private and shared between several authorities, which can not open the ballots without al least t shares of the decryption key. Then as long as a set of $p - t + 1$ authorities remains honest, the ballots cannot be opened before the end of the election.

Conclusion

In this thesis, we presented two new cryptographic constructions, which are a group signature scheme and a blind signature scheme, along with an application to an electronic voting system. Our constructions are built upon the lattice-based cryptography, which is assumed to be post-quantum resistant and that have been appealing for its numerous advantages. Indeed, since the call of the NIST competition in 2016, an effervescence around the post-quantum cryptography has been observed, and among the different propositions that are presented during this competition, the most represented candidate is lattice-based cryptography.

The reason is that, firstly its security is based on very well understood worst-case average-case reductions that give a concrete security for each construction. Secondly its efficiency, which was very bad at the beginning, is currently reaching those of its counterparts based on "classical" cryptography. Finally some very interesting constructions, who were only dreamed of several years ago, have been reached thanks to the lattice-based cryptography, for example, to the best of our knowledge, there exists only one fully-homomorphic encryption scheme based on lattices by Gentry [Gen09]. Moreover, the lattices benefit a structure that allows them to build some primitives (like attribute-based encryption, fully homomorphic encryption) that are currently not been developed in the other post-quantum cryptosystems. This shows that the maturity of the lattice-based cryptography is the most advanced among the post-quantum candidates.

Group signature scheme. Our first contribution is a lattice-based group signature scheme with forward security and proven secure in the standard model. The group signature schemes are a very interesting cryptographic primitive, allowing anyone to anonymously prove that he belongs to a given group. Moreover group signatures on lattices is a very intensive topic of research, since more than 15 constructions have been published since the former proposition by Gordon et al. [GKV10]. The reason behind this attractiveness is that the efficiency of lattice-based group signature is far from their counterparts on classical cryptography. In fact, the topic of group signature is closely related to the topic of zero-knowledge proof, which is usually the main tool used in such constructions

and is also an intense subject of research.

This is why it is interesting to explore other ways to build a group signature scheme to overcome these limitations. In this thesis, we took as a basis the Kastumata and Yamada [KY19] construction, that departs of the usual framework by replacing the zero-knowledge proofs with an attribute-based signatures. However it is hard to compare the efficiency of the two frameworks, since there is no implementation of the second one. In addition to this lack of implementation, it would be interesting to replace the complexity leveraging, that we also make use, with an other method to prove the security of the scheme. The attribute-based signature developed in [KY19] can be a good alternative but then it is the size dependence in the number of group members that is limiting. Finally, beyond the forward security property, that we integrate in our construction, other properties like the dynamic one would be interesting to equip the [KY19] framework with.

Blind signature scheme. The second contribution we bring in this thesis, is a lattice-based blind signature scheme. The idea of this primitive is to allow any user to generate a signature on a message of its choice in an interactive way with a signer possessing the signing secret key. Originally such tools were designed for the e-cash systems, where any user can interact with an authority to generate digital money, but the user does not want his payment to be traced went spending the electronic coin. Later the blind signature were also used to design electronic vote as we have seen in the last chapter of this thesis.

While the lattice-based blind signatures constructions were not a trending topic after the former construction of Rückert [Rüc10], later it found a renewal of attractiveness with a lot of recent constructions [Pap+19; ABB19; ABB20; Hau+20]. However, the last paper by Hauck et al. [Hau+20] found some serious issues in all the previous constructions, that are related the one-more unforgeability security. While their result proposes to fix this problem, their construction remains far from efficient. In our contribution we tried to reconcile the efficiency aspect with the security one, indeed we propose a slightly different approach, but we have to rely on a conjecture to prove the security of our scheme. Then it would be nice to succeed in proving this conjecture or find another way to prove the corresponding security. We can also denote that our construction needs to use the costly noise flooding to ensure the blindness property, finding an alternative would also be great.

Electronic voting system. Our last contribution concerns an electronic voting scheme that is based on the above blind signature scheme. It starts from the original framework

of Fujioka et al. [FOO92], and integrates some modifications like the replacement of the commitment scheme with an encryption scheme, or the transformation of the cryptographic tools into threshold variants. This framework departs from the usual electronic voting systems that make use of the advantage brought by homomorphic primitives. The consequence of such a modification is that our system is more efficient in some way, like the verification that a ballot is valid, while some other are less efficient, like the tally of the election.

In our e-voting construction, the blind signature is the main component, meaning that the features of our voting system are closely related on those of the blind signature, meaning that the improvement of the electronic voting system implies the improvements of the blind signature that have been listed above. Moreover the next task, that is currently considered in this contribution, is to provide an implementation to clearly analyze how this system can behave in a real-world election.

BIBLIOGRAPHY

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen, « Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE », in: *CRYPTO*, vol. 6223, Lecture Notes in Computer Science, Springer, 2010, pp. 98–115.

[ABB19] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann, « BLAZE: Practical Lattice-Based Blind Signatures for Privacy-Preserving Applications », in: *IACR Cryptology ePrint Archive* 2019 (2019), Version 20200207:124758, p. 1167.

[ABB20] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann, *On Lattice-Based Interactive Protocols with Aborts*, Cryptology ePrint Archive, Report 2020/007, `https://eprint.iacr.org/2020/007`, 2020.

[Adi08] Ben Adida, « Helios: Web-based Open-Audit Voting », in: *USENIX Security Symposium*, USENIX Association, 2008, pp. 335–348.

[AF96] Masayuki Abe and Eiichiro Fujisaki, « How to Date Blind Signatures », in: *ASIACRYPT*, vol. 1163, Lecture Notes in Computer Science, Springer, 1996, pp. 244–251.

[Agr+13] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai, « Discrete Gaussian Leftover Hash Lemma over Infinite Domains », in: *ASIACRYPT (1)*, vol. 8269, Lecture Notes in Computer Science, Springer, 2013, pp. 97–116.

[Ajt96] Miklós Ajtai, « Generating Hard Instances of Lattice Problems (Extended Abstract) », in: *STOC*, ACM, 1996, pp. 99–108.

[Alk+16] Erdem Alkım, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe, « Post-quantum Key Exchange - A New Hope », in: *USENIX Security Symposium*, USENIX Association, 2016, pp. 327–343.

[AO00] Masayuki Abe and Tatsuaki Okamoto, « Provably Secure Partially Blind Signatures », in: *CRYPTO*, vol. 1880, Lecture Notes in Computer Science, Springer, 2000, pp. 271–286.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott, « On the concrete hardness of Learning with Errors », in: *J. Mathematical Cryptology* 9.*3* (2015), pp. 169–203.

[AT99] Giuseppe Ateniese and Gene Tsudik, « Group Signatures *Á la carte* », in: *SODA*, ACM/SIAM, 1999, pp. 848–849.

[Ate+00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik, « A Practical and Provably Secure Coalition-Resistant Group Signature Scheme », in: *CRYPTO*, vol. 1880, Lecture Notes in Computer Science, Springer, 2000, pp. 255–270.

[AW04] Michel Abdalla and Bogdan Warinschi, « On the Minimal Assumptions of Group Signature Schemes », in: *Information and Communications Security*, vol. 3269, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 1–13.

[Ban93] W. Banaszczyk, « New bounds in some transference theorems in the geometry of numbers. », in: *Mathematische Annalen* 296.*4* (1993), pp. 625–636, URL: `http://eudml.org/doc/165105`.

[Bau+16] Carsten Baum, Ivan Damgård, Sabine Oechsner, and Chris Peikert, « Efficient Commitments and Zero-Knowledge Protocols from Ring-SIS with Applications to Lattice-based Threshold Cryptosystems », in: *IACR Cryptol. ePrint Arch.* 2016 (2016), p. 997.

[BB04] Dan Boneh and Xavier Boyen, « Secure Identity Based Encryption Without Random Oracles », in: *CRYPTO*, vol. 3152, Lecture Notes in Computer Science, Springer, 2004, pp. 443–459.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham, « Short Group Signatures », in: *CRYPTO*, vol. 3152, Lecture Notes in Computer Science, Springer, 2004, pp. 41–55.

[BCN18] Cecilia Boschini, Jan Camenisch, and Gregory Neven, « Floppy-Sized Group Signatures from Lattices », in: *ACNS*, vol. 10892, Lecture Notes in Computer Science, Springer, 2018, pp. 163–182.

[Bel+98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway, « Relations Among Notions of Security for Public-Key Encryption Schemes », in: *CRYPTO*, vol. 1462, Lecture Notes in Computer Science, Springer, 1998, pp. 26–45.

[Ber+15] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi, « SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions », in: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2015, pp. 499–516.

[Ber+19] Pauline Bert, Gautier Eberhart, Adeline Roux-Langlois, and Mohamed Sabt, « Implementation of Lattice Trapdoors on Modules and Applications to Signature », in: Private communication between authors, 2019.

[BF11] Dan Boneh and David Mandell Freeman, « Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures », in: *Public Key Cryptography*, vol. 6571, Lecture Notes in Computer Science, Springer, 2011, pp. 1–16.

[BKP13] Rikke Bendlin, Sara Krehbiel, and Chris Peikert, « How to Share a Lattice Trapdoor: Threshold Protocols for Signatures and (H)IBE », in: *ACNS*, vol. 7954, Lecture Notes in Computer Science, Springer, 2013, pp. 218–236.

[Bla+17] Olivier Blazy, Philippe Gaborit, Julien Schrek, and Nicolas Sendrier, « A code-based blind signature », in: *ISIT*, IEEE, 2017, pp. 2718–2722.

[BM93] Josh Cohen Benaloh and Michael de Mare, « One-Way Accumulators: A Decentralized Alternative to Digital Sinatures (Extended Abstract) », in: *EUROCRYPT*, vol. 765, Lecture Notes in Computer Science, Springer, 1993, pp. 274–285.

[BM99] Mihir Bellare and Sara K. Miner, « A Forward-Secure Digital Signature Scheme », in: *CRYPTO*, vol. 1666, Lecture Notes in Computer Science, Springer, 1999, pp. 431–448.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi, « Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions », in: *EUROCRYPT*, vol. 2656, Lecture Notes in Computer Science, Springer, 2003, pp. 614–629.

[BN06] Mihir Bellare and Gregory Neven, « Multi-signatures in the plain public-Key model and a general forking lemma », in: *ACM Conference on Computer and Communications Security*, ACM, 2006, pp. 390–399.

[Boy+06] Xavier Boyen, Hovav Shacham, Emily Shen, and Brent Waters, « Forward-secure signatures with untrusted update », in: *ACM Conference on Computer and Communications Security*, ACM, 2006, pp. 191–200.

[BR93] Mihir Bellare and Phillip Rogaway, « Random Oracles are Practical: A Paradigm for Designing Efficient Protocols », in: *ACM Conference on Computer and Communications Security*, ACM, 1993, pp. 62–73.

[Bra93] Stefan Brands, « Untraceable Off-line Cash in Wallets with Observers (Extended Abstract) », in: *CRYPTO*, vol. 773, Lecture Notes in Computer Science, Springer, 1993, pp. 302–318.

[BS04] Dan Boneh and Hovav Shacham, « Group signatures with verifier-local revocation », in: *ACM Conference on Computer and Communications Security*, ACM, 2004, pp. 168–177.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang, « Foundations of Group Signatures: The Case of Dynamic Groups », in: *CT-RSA*, vol. 3376, Lecture Notes in Computer Science, Springer, 2005, pp. 136–153.

[BW06] Xavier Boyen and Brent Waters, « Compact Group Signatures Without Random Oracles », in: *EUROCRYPT*, vol. 4004, Lecture Notes in Computer Science, Springer, 2006, pp. 427–444.

[BY03] Mihir Bellare and Bennet S. Yee, « Forward-Security in Private-Key Cryptography », in: *CT-RSA*, vol. 2612, Lecture Notes in Computer Science, Springer, 2003, pp. 1–18.

[Cam+20] Jan Camenisch, Manu Drijvers, Anja Lehmann, Gregory Neven, and Patrick Towa, « Short Threshold Dynamic Group Signatures », in: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 16.

[Cam97] Jan Camenisch, « Efficient and Generalized Group Signatures », in: *EUROCRYPT*, vol. 1233, Lecture Notes in Computer Science, Springer, 1997, pp. 465–479.

[Can01] Ran Canetti, « Universally Composable Security: A New Paradigm for Cryptographic Protocols », in: *FOCS*, IEEE Computer Society, 2001, pp. 136–145.

[Cas+10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert, « Bonsai Trees, or How to Delegate a Lattice Basis », in: *EUROCRYPT*, vol. 6110, Lecture Notes in Computer Science, Springer, 2010, pp. 523–552.

[CGT06] Sébastien Canard, Matthieu Gaud, and Jacques Traoré, « Defeating Malicious Servers in a Blind Signatures Based Voting System », in: *Financial Cryptography*, vol. 4107, Lecture Notes in Computer Science, Springer, 2006, pp. 148–153.

[CH91] David Chaum and Eugène van Heyst, « Group Signatures », in: *EUROCRYPT*, vol. 547, Lecture Notes in Computer Science, Springer, 1991, pp. 257–265.

[Cha82] David Chaum, « Blind Signatures for Untraceable Payments », in: *CRYPTO*, Plenum Press, New York, 1982, pp. 199–203.

[Che+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé, « Cryptanalysis of the Multilinear Map over the Integers », in: *EUROCRYPT (1)*, vol. 9056, Lecture Notes in Computer Science, Springer, 2015, pp. 3–12.

[Chi+16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène, « A Homomorphic LWE Based E-voting Scheme », in: *PQCrypto*, vol. 9606, Lecture Notes in Computer Science, Springer, 2016, pp. 245–265.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz, « A Forward-Secure Public-Key Encryption Scheme », in: *EUROCRYPT*, vol. 2656, Lecture Notes in Computer Science, Springer, 2003, pp. 255–271.

[CL04] Jan Camenisch and Anna Lysyanskaya, « Signature Schemes and Anonymous Credentials from Bilinear Maps », in: *CRYPTO*, vol. 3152, Lecture Notes in Computer Science, Springer, 2004, pp. 56–72.

[CMM19] Núria Costa, Ramiro Martínez, and Paz Morillo, « Lattice-Based Proof of a Shuffle », in: *Financial Cryptography Workshops*, vol. 11599, Lecture Notes in Computer Science, Springer, 2019, pp. 330–346.

[CNR12] Jan Camenisch, Gregory Neven, and Markus Rückert, « Fully Anonymous Attribute Tokens from Lattices », in: *SCN*, vol. 7485, Lecture Notes in Computer Science, Springer, 2012, pp. 57–75.

[Cor+13] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène, « A generic construction for voting correctness at minimum cost - Application to Helios », in: *IACR Cryptol. ePrint Arch.* 2013 (2013), p. 177.

[Cor+14] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène, « Election Verifiability for Helios under Weaker Trust Assumptions », in: *ESORICS (2)*, vol. 8713, Lecture Notes in Computer Science, Springer, 2014, pp. 327–344.

[Cor+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi, « Zeroizing Without Low-Level Zeroes: New MMAP Attacks and their Limitations », in: *CRYPTO (1)*, vol. 9215, Lecture Notes in Computer Science, Springer, 2015, pp. 247–266.

[Cor+16] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung, « SoK: Verifiability Notions for E-Voting Protocols », in: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2016, pp. 779–798.

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve A. Schneider, « A Practical Voter-Verifiable Election Scheme », in: *ESORICS*, vol. 3679, Lecture Notes in Computer Science, Springer, 2005, pp. 118–139.

[CS18] Remi Clarisse and Olivier Sanders, « Short Group Signature in the Standard Model », in: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 1115.

[DOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener, « Authentication and Authenticated Key Exchanges », in: *Des. Codes Cryptogr.* 2.*2* (1992), pp. 107–125.

[Fer93] Niels Ferguson, « Single Term Off-Line Coins », in: *EUROCRYPT*, vol. 765, Lecture Notes in Computer Science, Springer, 1993, pp. 318–328.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto, « How to Enhance the Security of Public-Key Encryption at Minimum Cost », in: *Public Key Cryptography*, vol. 1560, Lecture Notes in Computer Science, Springer, 1999, pp. 53–68.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta, « A Practical Secret Voting Scheme for Large Scale Elections », in: *AUSCRYPT*, vol. 718, Lecture Notes in Computer Science, Springer, 1992, pp. 244–251.

[FS86] Amos Fiat and Adi Shamir, « How to Prove Yourself: Practical Solutions to Identification and Signature Problems », in: *CRYPTO*, vol. 263, Lecture Notes in Computer Science, Springer, 1986, pp. 186–194.

[Gen09] Craig Gentry, « Fully homomorphic encryption using ideal lattices », in: *STOC*, ACM, 2009, pp. 169–178.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi, « Candidate Multilinear Maps from Ideal Lattices », in: *EUROCRYPT*, vol. 7881, Lecture Notes in Computer Science, Springer, 2013, pp. 1–17.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi, « Public-Key Cryptosystems from Lattice Reduction Problems », in: *CRYPTO*, vol. 1294, Lecture Notes in Computer Science, Springer, 1997, pp. 112–131.

[GKV10] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan, « A Group Signature Scheme from Lattice Assumptions », in: *ASIACRYPT*, vol. 6477, Lecture Notes in Computer Science, Springer, 2010, pp. 395–412.

[GM18] Nicholas Genise and Daniele Micciancio, « Faster Gaussian Sampling for Trapdoor Lattices with Arbitrary Modulus », in: *EUROCRYPT (1)*, vol. 10820, Lecture Notes in Computer Science, Springer, 2018, pp. 174–203.

[Gol+10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan, « Robustness of the Learning with Errors Assumption », in: *ICS*, Tsinghua University Press, 2010, pp. 230–240.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan, « Trapdoors for hard lattices and new cryptographic constructions », in: *STOC*, ACM, 2008, pp. 197–206.

[Gro07] Jens Groth, « Fully Anonymous Group Signatures Without Random Oracles », in: *ASIACRYPT*, vol. 4833, Lecture Notes in Computer Science, Springer, 2007, pp. 164–180.

[Gün89] Christoph G. Günther, « An Identity-Based Key-Exchange Protocol », in: *EUROCRYPT*, vol. 434, Lecture Notes in Computer Science, Springer, 1989, pp. 29–37.

[Hås+99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby, « A Pseudorandom Generator from any One-way Function », in: *SIAM J. Comput.* 28.*4* (1999), pp. 1364–1396.

[Hau+20] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen, « Lattice-Based Blind Signatures, Revisited », in: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 769.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz, « A Modular Analysis of the Fujisaki-Okamoto Transformation », in: *TCC (1)*, vol. 10677, Lecture Notes in Computer Science, Springer, 2017, pp. 341–371.

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss, « A Modular Treatment of Blind Signatures from Identification Schemes », in: *EUROCRYPT (3)*, vol. 11478, Lecture Notes in Computer Science, Springer, 2019, pp. 345–375.

[IR01] Gene Itkis and Leonid Reyzin, « Forward-Secure Signatures with Optimal Signing and Verifying », in: *CRYPTO*, vol. 2139, Lecture Notes in Computer Science, Springer, 2001, pp. 332–354.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky, « Security of Blind Digital Signatures (Extended Abstract) », in: *CRYPTO*, vol. 1294, Lecture Notes in Computer Science, Springer, 1997, pp. 150–164.

[KR05] Steve Kremer and Mark Ryan, « Analysis of an Electronic Voting Protocol in the Applied Pi Calculus », in: *ESOP*, vol. 3444, Lecture Notes in Computer Science, Springer, 2005, pp. 186–200.

[KW20] Sam Kim and David J. Wu, « Multi-theorem Preprocessing NIZKs from Lattices », in: *Journal of Cryptology* 33.*3* (2020), pp. 619–702.

[KY19] Shuichi Katsumata and Shota Yamada, « Group Signatures Without NIZK: From Lattices in the Standard Model », in: *EUROCRYPT (3)*, vol. 11478, Lecture Notes in Computer Science, Springer, 2019, pp. 312–344.

[Lag+13] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé, « Lattice-Based Group Signatures with Logarithmic Signature Size », in: *ASIACRYPT (2)*, vol. 8270, Lecture Notes in Computer Science, Springer, 2013, pp. 41–61.

[Lan+14] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang, « Lattice-Based Group Signature Scheme with Verifier-Local Revocation », in: *Public Key Cryptography*, vol. 8383, Lecture Notes in Computer Science, Springer, 2014, pp. 345–361.

[Lib+16a] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang, « Signature Schemes with Efficient Protocols and Dynamic Group Signatures from Lattice Assumptions », in: *ASIACRYPT (2)*, vol. 10032, Lecture Notes in Computer Science, 2016, pp. 373–403.

[Lib+16b] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang, « Zero-Knowledge Arguments for Lattice-Based Accumulators: Logarithmic-Size Ring Signatures and Group Signatures Without Trapdoors », in: *EUROCRYPT (2)*, vol. 9666, Lecture Notes in Computer Science, Springer, 2016, pp. 1–31.

[Lin+13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang, « Improved Zero-Knowledge Proofs of Knowledge for the ISIS Problem, and Applications », in: *Public Key Cryptography*, vol. 7778, Lecture Notes in Computer Science, Springer, 2013, pp. 107–124.

[Lin+14] San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld, « Hardness of k-LWE and Applications in Traitor Tracing », in: *CRYPTO (1)*, vol. 8616, Lecture Notes in Computer Science, Springer, 2014, pp. 315–334.

[Lin+17] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu, « Lattice-Based Group Signatures: Achieving Full Dynamicity with Ease », in: *ACNS*, vol. 10355, Lecture Notes in Computer Science, Springer, 2017, pp. 293–312.

[Lin+19] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu, « Forward-Secure Group Signatures from Lattices », in: *PQCrypto*, vol. 11505, Lecture Notes in Computer Science, Springer, 2019, pp. 44–64.

[LM06] Vadim Lyubashevsky and Daniele Micciancio, « Generalized Compact Knapsacks Are Collision Resistant », in: *ICALP (2)*, vol. 4052, Lecture Notes in Computer Science, Springer, 2006, pp. 144–155.

[LMN16] Benoît Libert, Fabrice Mouhartem, and Khoa Nguyen, « A Lattice-Based Group Signature Scheme with Message-Dependent Opening », in: *ACNS*, vol. 9696, Lecture Notes in Computer Science, Springer, 2016, pp. 137–155.

[LNW15] San Ling, Khoa Nguyen, and Huaxiong Wang, « Group Signatures from Lattices: Simpler, Tighter, Shorter, Ring-Based », in: *Public Key Cryptography*, vol. 9020, Lecture Notes in Computer Science, Springer, 2015, pp. 427–449.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev, « On Ideal Lattices and Learning with Errors over Rings », in: *EUROCRYPT*, vol. 6110, Lecture Notes in Computer Science, Springer, 2010, pp. 1–23.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev, « A Toolkit for Ring-LWE Cryptography », in: *EUROCRYPT*, vol. 7881, Lecture Notes in Computer Science, Springer, 2013, pp. 35–54.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld, « GGHLite: More Efficient Multilinear Maps from Ideal Lattices », in: *EUROCRYPT*, vol. 8441, Lecture Notes in Computer Science, Springer, 2014, pp. 239–256.

[LV09] Benoît Libert and Damien Vergnaud, « Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model », in: *CANS*, vol. 5888, Lecture Notes in Computer Science, Springer, 2009, pp. 498–517.

[LY10] Benoît Libert and Moti Yung, « Dynamic fully forward-secure group signatures », in: *AsiaCCS*, ACM, 2010, pp. 70–81.

[Lyu08] Vadim Lyubashevsky, « Lattice-Based Identification Schemes Secure Under Active Attacks », in: *Public Key Cryptography*, vol. 4939, Lecture Notes in Computer Science, Springer, 2008, pp. 162–179.

[Lyu09] Vadim Lyubashevsky, « Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures », in: *ASIACRYPT*, vol. 5912, Lecture Notes in Computer Science, Springer, 2009, pp. 598–616.

[Lyu12] Vadim Lyubashevsky, « Lattice Signatures without Trapdoors », in: *EUROCRYPT*, vol. 7237, Lecture Notes in Computer Science, Springer, 2012, pp. 738–755.

[Moh10] Payman Mohassel, « One-Time Signatures and Chameleon Hash Functions », in: *Selected Areas in Cryptography*, vol. 6544, Lecture Notes in Computer Science, Springer, 2010, pp. 302–319.

[MP12] Daniele Micciancio and Chris Peikert, « Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller », in: *EUROCRYPT*, vol. 7237, Lecture Notes in Computer Science, Springer, 2012, pp. 700–718.

[MR07] Daniele Micciancio and Oded Regev, « Worst-Case to Average-Case Reductions Based on Gaussian Measures », in: *SIAM J. Comput.* 37.*1* (2007), pp. 267–302.

[MV03] Daniele Micciancio and Salil P. Vadhan, « Statistical Zero-Knowledge Proofs with Efficient Provers: Lattice Problems and More », in: *CRYPTO*, vol. 2729, Lecture Notes in Computer Science, Springer, 2003, pp. 282–298.

[NF05] Toru Nakanishi and Nobuo Funabiki, « Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps », in: *ASIACRYPT*, vol. 3788, Lecture Notes in Computer Science, Springer, 2005, pp. 533–548.

[NHF09] Toru Nakanishi, Yuta Hira, and Nobuo Funabiki, « Forward-Secure Group Signatures from Pairings », in: *Pairing*, vol. 5671, Lecture Notes in Computer Science, Springer, 2009, pp. 171–186.

[NZZ15] Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang, « Simpler Efficient Group Signatures from Lattices », in: *Public Key Cryptography*, vol. 9020, Lecture Notes in Computer Science, Springer, 2015, pp. 401–426.

[Oht+09] Go Ohtake, Arisa Fujii, Goichiro Hanaoka, and Kazuto Ogawa, « On the Theoretical Gap between Group Signatures with and without Unlinkability », in: *AFRICACRYPT*, vol. 5589, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 149–166.

[Oka92] Tatsuaki Okamoto, « Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes », in: *CRYPTO*, vol. 740, Lecture Notes in Computer Science, Springer, 1992, pp. 31–53.

[Pap+19] Dimitrios Papachristoudis, Dimitrios Hristu Varsakelis, Foteini Baldimtsi, and George Stephanides, « Leakage-resilient lattice-based partially blind signatures », in: *IET Information Security* 13.*6* (2019), pp. 670–684.

[Pet97] Holger Petersen, « How to Convert any Digital Signature Scheme into a Group Signature Scheme », in: *Security Protocols Workshop*, vol. 1361, Lecture Notes in Computer Science, Springer, 1997, pp. 177–190.

[Pin+17] Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler, « Practical Quantum-Safe Voting from Lattices », in: *ACM Conference on Computer and Communications Security*, ACM, 2017, pp. 1565–1581.

[PLS18] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler, « Lattice-Based Group Signatures and Zero-Knowledge Proofs of Automorphism Stability », in: *ACM Conference on Computer and Communications Security*, ACM, 2018, pp. 574–591.

[PR06] Chris Peikert and Alon Rosen, « Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices », in: *TCC*, vol. 3876, Lecture Notes in Computer Science, Springer, 2006, pp. 145–166.

[PS00] David Pointcheval and Jacques Stern, « Security Arguments for Digital Signatures and Blind Signatures », in: *J. Cryptology* 13.*3* (2000), pp. 361–396.

[PS19] Chris Peikert and Sina Shiehian, « Noninteractive Zero Knowledge for NP from (Plain) Learning with Errors », in: *CRYPTO (1)*, vol. 11692, Lecture Notes in Computer Science, Springer, 2019, pp. 89–114.

[PS96a] David Pointcheval and Jacques Stern, « Provably Secure Blind Signature Schemes », in: *ASIACRYPT*, vol. 1163, Lecture Notes in Computer Science, Springer, 1996, pp. 252–265.

[PS96b] David Pointcheval and Jacques Stern, « Security Proofs for Signature Schemes », in: *EUROCRYPT*, vol. 1070, Lecture Notes in Computer Science, Springer, 1996, pp. 387–398.

[PSM17] Albrecht Petzoldt, Alan Szepieniec, and Mohamed Saied Emam Mohamed, « A Practical Multivariate Blind Signature Scheme », in: *Financial Cryptography*, vol. 10322, Lecture Notes in Computer Science, Springer, 2017, pp. 437–454.

[Reg05] Oded Regev, « On lattices, learning with errors, random linear codes, and cryptography », in: *STOC*, ACM, 2005, pp. 84–93.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, « A Method for Obtaining Digital Signatures and Public-Key Cryptosystems », in: *Commun. ACM* 21.*2* (1978), pp. 120–126.

[Rüc10] Markus Rückert, « Lattice-Based Blind Signatures », in: *ASIACRYPT*, vol. 6477, Lecture Notes in Computer Science, Springer, 2010, pp. 413–430.

[San+18] San, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu, « Constant-Size Group Signatures from Lattices », in: *Public Key Cryptography (2)*, vol. 10770, Lecture Notes in Computer Science, Springer, 2018, pp. 58–88.

[Sch89] Claus-Peter Schnorr, « Efficient Identification and Signatures for Smart Cards », in: *CRYPTO*, vol. 435, Lecture Notes in Computer Science, Springer, 1989, pp. 239–252.

[Sha79] Adi Shamir, « How to Share a Secret », in: *Commun. ACM* 22.*11* (1979), pp. 612–613.

[Sho97] Peter W. Shor, « Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer », in: *SIAM J. Comput.* 26.*5* (1997), pp. 1484–1509.

[Son01] Dawn Xiaodong Song, « Practical forward secure group signature schemes », in: *ACM Conference on Computer and Communications Security*, ACM, 2001, pp. 225–234.

[Ste+09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa, « Efficient Public Key Encryption Based on Ideal Lattices », in: *ASIACRYPT*, vol. 5912, Lecture Notes in Computer Science, Springer, 2009, pp. 617–635.

[Tsa17] Rotem Tsabary, « An Equivalence Between Attribute-Based Signatures and Homomorphic Signatures, and New Constructions for Both », in: *TCC (2)*, vol. 10678, Lecture Notes in Computer Science, Springer, 2017, pp. 489–518.

[TZW16] Haibo Tian, Fangguo Zhang, and Baodian Wei, « A lattice-based partially blind signature », in: *Security and Communication Networks* 9.*12* (2016), pp. 1820–1828.

[Yue+12] Tsz Hon Yuen, Joseph K. Liu, Xinyi Huang, Man Ho Au, Willy Susilo, and Jianying Zhou, « Forward Secure Attribute-Based Signatures », in: *ICICS*, vol. 7618, Lecture Notes in Computer Science, Springer, 2012, pp. 167–177.

[Zha+18] Pingyuan Zhang, Han Jiang, Zhihua Zheng, Peichu Hu, and Qiuliang Xu, « A New Post-Quantum Blind Signature From Lattice Assumptions », in: *IEEE Access* 6 (2018), pp. 27251–27258.

[Zha+19] Yanhua Zhang, Ximeng Liu, Yupu Hu, Qikun Zhang, and Huiwen Jia, « Lattice-Based Group Signatures with Verifier-Local Revocation: Achieving Shorter Key-Sizes and Explicit Traceability with Ease », in: *CANS*, vol. 11829, Lecture Notes in Computer Science, Springer, 2019, pp. 120–140.

UNIVERSITÉ DE RENNES 1

Titre : Cryptographie post-quantique pour la protection de la vie privée

Mot clés : Cryptographie, Signatures, Réseaux euclidiens, Vote électronique

Résumé : Ces dernières années la cryptographie a été chamboulée par l'arrivée des ordinateurs quantiques. En effet ceux-ci possèdent un très fort avantage pour casser les schémas cryptographique utilisés actuellement dans la quasi-totalité des communications séurisées.
Nous proposons dans cette thèse plusieurs constructions cryptographiques basées sur des outils mathématiques résistants à ces ordinateurs quantique, que sont les réseaux euclidiens.
Tout d'abord nous construisons une signature de groupe, permettant à chaque membre composant un groupe donné de signer au nom du groupe tout en conservant son anynonymat. Nous rajoutons une propriété supplémentaire qui est la *forward secrecy* qui sépare le temps en périodes durant lesquelles les clés secrètes des utilisateurs sont mises à jour.
Nous proposons également un schéma de signature aveugle qui permet à un utilisateur de générer une signature sur un message de son choix de manière interactive avec un signeur qui possède la clé de signature. Nous améliorons l'état-de-l'art en proposant un schéma sans abandon et avec une sécurité plus efficace.
Enfin, comme cas d'usage de la signature aveugle nous construisons un schéma de vote électronique à partir de cette primitive.

Title: Lattice-based cryptography for privacy

Keywords: Cryptography, Signatures, Lattices, Evote

Abstract: The past few years have seen the rising of the quantum computers, that are a serious threat to nearly all the actual cryptographic schemes used in practice.
In this thesis we propose some new constructions to prevent this obsolescence by building our schemes on the mathematical tool of lattices that is assumed post-quantum resistant.
We firstly develop a group signature scheme, allowing each member composing the group to anonymously sign on the behalf of the group. We add a supplementary property, which is the froward secrecy. This property cut the time in periods, such that each secret key is updated when entering a new period.
We also propose a blind signature scheme, which is an interactive protocol between an user, who wants to sign a message, with a signer who possesses the signing secret key. We improve the state-of-the art by proposing a constructions without any restart and with a more efficient security.
Finally as a use case of the blind signature, we develop an evoting protocol that take as a basis the construction described above.