PhD-FSTC-2014-15
The Faculty of Sciences, Technology
and Communication

Numéro d'enregistrement STEP : 79787
École Normale Supérieure
École Doctorale 386 – Sciences
mathématiques de Paris Centre
UMR 8548 Laboratoire d'informatique
de l'École Normale Supérieure – LIENS

DISSERTATION

Defense held on 30/06/2014 in Paris, France

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG EN INFORMATIQUE

AND

DOCTEUR DE L'ÉCOLE NORMALE SUPÉRIEURE EN INFORMATIQUE

by

Tancrède LEPOINT
Born on 23 October 1988 in Seclin (France)

DESIGN AND IMPLEMENTATION OF LATTICE-BASED CRYPTOGRAPHY

Dissertation defense committee

Dr Jean-Sébastien Coron, dissertation supervisor
Assistant Professor, Université du Luxembourg

Dr David Naccache
Professor, Université Paris II, Panthéon-Assas, École Normale Supérieure

Dr David Pointcheval, dissertation supervisor
CNRS, École Normale Supérieure

Dr Peter Y. D. Ryan, Chairman
Professor, Université du Luxembourg

Dr Damien Stehlé, reviewer
Professor, École Normale Supérieure de Lyon

Dr Vinod Vaikuntanathan, reviewer
Assistant Professor, Massachusetts Institute of Technology

Remerciements

Mes premiers remerciements vont à mes directeurs de thèse Jean-Sébastien Coron et David Pointcheval. Merci à Jean-Sébastien dont les compétences larges et avisées en cryptologie, la rigueur scientifique et les idées clairvoyantes m'ont beaucoup appris et permis d'appréhender la vastité de la cryptographie. Son encadrement de qualité et son enthousiasme ont incontestablement eu un impact positif sur ma recherche. Je suis également très reconnaissant à David pour m'avoir bienveillamment accueilli dans l'équipe de cryptologie de l'École Normale Supérieure, et pour toutes nos discussions intéressantes autour d'un tableau blanc ou de fraises au chocolat qui ont façonné mes visions de la cryptographie et de la recherche académique.

J'exprime maintenant ma plus grande gratitude envers Pascal Paillier, qui m'a donné l'opportunité d'effectuer ma thèse en milieu industriel dans les meilleures conditions possibles, a partagé avec moi son expertise pointue dans tous les domaines de la cryptologie et m'a soutenu dans toutes sortes de projets, qu'ils soient professionnels ou personnels. Son optimisme est une motivation constante à donner le meilleur de soi-même, ainsi qu'un plaisir à vivre.

I am endlessly grateful to Damien Stehlé and Vinod Vaikuntanathan who produced thoughtful and insightful reviews of my dissertation in spite of their busy schedule. I strongly admire their research work, and I feel extremely fortunate they accepted to read, comment and endorse my thesis. I would also like to say thank you to Peter Y. D. Ryan who kindly agreed to be in my committee, and to be the chairman. Merci également à David Naccache, qui a accepté de faire partie de mon jury de thèse, et qui n'hésite pas à partager avec passion ses idées, ses découvertes et ses anecdotes de vie.

Je tiens aussi à remercier également tous ceux qui m'ont fait des remarques constructives sur les versions préliminaires de chapitres de ce manuscrit : Damien, Pascal, David, Jean-Sébastien, Édouard et Thomas. Je remercie également tous mes co-auteurs : Jean-Sébastien, Mehdi, Marc, Matthieu, Pascal, Aaram, Alain, Bart, Cécile, Jinsu, Jung Hee, Léo, Michael, Moon Sung, Peter, Vadim et Yoni; ces nombreuses collaborations ont été riches d'enseignements et ont contribué à façonner ma manière d'écrire et de penser la recherche.

J'ai été particulièrement privilégié de faire ma thèse au sein de CryptoExperts (merci beaucoup Marc de m'avoir recommandé !), qui non seulement offre un environnement de travail idéal en présence de personnes dont les compétences et la gentillesse continuent de me frapper chaque jour davantage, mais aussi une très grande liberté qui m'a permis de découvrir le monde à maintes reprises. Merci donc à Antoine, Cécile, Louis, Matthieu F., Matthieu R., Pascal et Thomas pour ces trois années vraiment excellentes, et merci de m'offrir une place parmi vous pour la suite.

Je remercie également toute l'équipe de l'ENS pour l'ambiance toujours agréable au labo, la disponibilité des permanents, l'enthousiasme des stagiaires, l'aide entre doctorants, le melting pot de cultures et de nationalités. Tout cela amène des discussions passionnantes sur de nombreux sujets, que ce soit à propos de cryptographie, de recherche, de culture, et même de potins. Merci aux présents (Adrian, Alain, Angelo, Antonia, Céline, Damien, David N., David P., Duong, Fabrice, Florian, Geoffroy, Hoeteck, Houda, Itai, Mario, Michel, Olivier, Pierrick, Rafael, Sonia, Sylvain, Thomas et Vadim), aux temporairement absents (Jacques, Phong) et à ceux qui sont partis et que j'ai connus (Aurore, Dario, Elizabeth, Jérémy, Léo, Mario, Miriam, Nuttapong, Olivier, Patrick, Pierre-Alain, Sorina et Yuanmi).

Between my second and third year, I was really lucky to have the opportunity to carry out an internship at Microsoft Research in Redmond, with Kristin and Michael as mentors. Working there was a really fulfilling experience. Not only I learned a lot about cryptography, but I also got to discover the work in a big research company and the life in the U.S. for three months. I would like to thank the members of the Cryptography group (Craig, Joppe, Josh, Kristin, Melissa, Michael and Seny), and my fellow interns (Adriana, Alyson, Andrea, Joop, Kim and Sarah) for the really good time I had. Thanks also to Ben and Shane to whom I wish a successful life, professionally and personally.

Je suis très reconnaissant aussi à toutes ces rencontres professionnelles de ces dernières années qui rendent la communauté cryptographique attachante. Je pense notamment au soutien initial précieux de Grégory, Marc et Pascal, à la complicité avec Emmanuela, aux partages de nourriture (littéralement et intellectuellement) avec Hoeteck, à Sonia que je recroise au fil des études – avec plaisir – un peu de façon inopinée, à l'équipe des bristoliens (Emmanuela, Enrique, Jake, Joop, Gareth, Marcel, Peter, Valentina) que je retrouve un peu partout dans le monde, aux discussions en tête à tête avec des chercheurs réputés, venus d'ailleurs, qui trouvent néanmoins le temps de faire connaissance (Benoît, Chris, Damien, Dan, Douglas, Daniele, Henri, Jung Hee, Kenny, Nigel, Shai, Tanja, Thomas, Vinod, Xavier) et à mes collègues qui deviennent des amis précieux.

Je tiens aussi à remercier Adeline, Hugo, Julien, Peter et Xavier, thésards de 2011 comme moi, avec qui j'ai pu partager et comparer mon expérience de thèse (ses hauts et ses bas), et qui ont été un soutien sans doute plus important que ce qu'ils imaginent.

Merci aussi à tous ceux qui viennent et font partie de ma vie en dehors de mes études : mes parents qui m'ont toujours soutenu, mes frères et sœurs (César, Mathilde, Jeanne, Lucas, Solal et Raphaëlle) qui rendent la famille si bruyante mais précieuse, Jickye, Thomas, *Bow Tie* Mike, Anne-Sophie, Tom, Amandine et Hugo, Jill-Jênn, Shane, et enfin Laure, Clément et David qui ont une place toute particulière.

En dernier lieu, je remercie de tout mon cœur Édouard qui me rend profondément heureux à partager ma vie depuis plus de cinq ans.

Contents

Chapitre 1

Présentation des résultats et perspectives futures

Comme illustré par les travaux présentés dans ce manuscrit, l'objectif principal de notre recherche consiste à réduire l'écart entre la théorie et la pratique de la cryptographie à clé publique récente. En particulier, toutes nos nouvelles conceptions de schémas sont accompagnées d'implémentations destinées à valider les-dits résultats théoriques et permettent ainsi de donner un ordre de grandeur de leur potentiel en pratique. Notre thèse est divisée en trois parties, chacune se concentrant sur une primitive cryptographique différente.

Dans la première partie, nous décrivons un nouveau schéma de signature numérique dont la sécurité est basée sur des problèmes relatifs aux réseaux. La cryptographie à base de réseaux est réputée être « asymptotiquement efficace » mais toutes ses instanciations (prouvées sures[1]) utilisaient jusqu'alors des paramètres trop grands pour envisager leur utilisation pratique (en particulier sur petite architecture). Notre contribution principale consiste alors à décrire une nouvelle signature numérique compacte (avec des signatures de l'ordre de 5000 bits), performante, sure et adaptée aux environnements contraints. Les améliorations théoriques ont été combinées avec des optimisations pratiques et ont permis d'obtenir une signature numérique basée sur les réseaux aussi efficace (voire plus efficace) que celles reposant sur RSA ou sur les courbes elliptiques.

Dans la seconde partie, nous nous efforçons de rendre le chiffrement complètement homomorphe (parfois considéré comme le *Saint Graal* de la cryptographie [Mic10]) plus efficace. Ce dernier permet d'effectuer (de façon publique) des calculs arbitraires sur des messages chiffrés. Les premières instanciations de cette surprenante primitive ne peuvent être considérées comme pratiques, chaque multiplication de deux bits chiffrés nécessitant d'être suivie par une procédure de plusieurs dizaines de minutes [GH11b, CNT12]. Notre contribution principale consiste à améliorer les schémas complètement homomorphes sur les entiers [vDGHV10, CMNT11, CNT12] afin d'évaluer de façon homomorphe un circuit non trivial (nous avons choisi l'AES, comme Gentry, Halevi et Smart [GHS12c]). Une telle évaluation de l'AES avec nos schémas nécessite 102 heures mais permet de traiter 1875 blocs de 128 bits en parallèle, ce qui donne un temps relatif (par bloc) de 3 minutes, comparable au temps relatif de 5 minutes obtenu dans [GHS12c] en utilisant un schéma basé sur les réseaux [BGV12].

Dans la troisième partie, nous construisons des applications multilinéaires cryptographiques. Cette primitive, généralisant les « couplages » (applications bilinéaires), admet une première construction basée sur les réseaux depuis fin 2012 [GGH13a] et a de nombreuses conséquences inattendues et à fort potentiel (comme l'existence d'obfuscation indistinguable ou de chiffrement fonctionnel pour tous les circuits [GGH$^+$13b]). Présentée sans preuve de sécurité (ce qui est insolite dans la cryptographie à clé publique moderne), ses paramètres ont été choisis pour résister à toutes les attaques connues. Dans cette partie, nous proposons une nouvelle construction, similaire en essence mais différente en pratique, qui permet de donner une primitive alternative dont la sécurité (également heuristique) diffère. En particulier, notre primitive semble invulnérable à l'attaque par

[1] Dans ce chapitre, nous avons adopté les rectifications orthographiques du français de 1990 qui simplifient et suppriment certaines incohérences du français, entre autres en supprimant des accents circonflexes et traits d'union.

« zeroizing » [GGH13a] qui rend facile les analogues du problème décisionnel linéaire (DLIN) ou du problème d'appartenance à un sous-groupe dans la construction de Garg, Gentry et Halevi. Ainsi certaines applications nécessitant que ces problèmes soient difficiles, comme le chiffrement fonctionnel prouvé sûr contre les attaquants adaptatifs, les preuves de connaissance non interactives à divulgation nulle de connaissance ou les échanges de clés avec authentification par mot de passe robustes contre la corruption des serveurs [BP13], nécessitent d'utiliser notre construction. Finalement, nous décrivons la première implémentation « preuve de concept » d'une telle primitive (la construction initiale n'étant que théorique) et montrons qu'un échange de clé entre 7 participants nécessite moins d'une quarantaine de secondes sur un processeur classique actuel en utilisant nos applications multilinéaires cryptographiques.

1.1 Contributions à la cryptographie basée sur les réseaux [DDLL13a]

Bien qu'utilisés traditionnellement en cryptanalyse [LLL82, SE94] depuis 1982, ce n'est qu'une quinzaine d'années plus tard que les réseaux ont fait l'objet d'une découverte en tant qu'outil de conception de schémas cryptographiques (Ajtai-Dwork [Ajt96], GGH [GGH97], NTRU [HPS98]). Tout en ouvrant la voie à de nouvelles primitives à clé publique, la cryptographie à base de réseaux s'illustre par certaines qualités très attrayantes, comme une sécurité basée sur des problèmes dans le pire cas (beaucoup étudiés dans d'autres domaines) et une simplicité d'implémentation grâce aux opérations élémentaires simples sous-jacentes (multiplication de vecteurs et matrices à coefficients entiers).

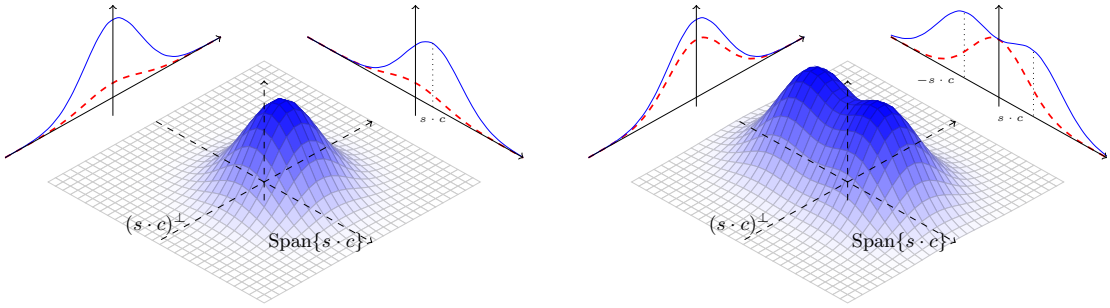
Malheureusement, cette simplicité apparente s'aheurte à la compacité souhaitée en pratique. En particulier, si les complexités asymptotiques des schémas basés sur les réseaux peuvent être quasi-optimales, les paramètres permettant d'atteindre des niveaux cibles de sécurité classiques (disons de 80 à 256 bits) produisent des clés, chiffrés et signatures de grande taille. De plus les schémas admettant des preuves de sécurité utilisent le plus souvent des Gaussiennes discrètes, dont l'échantillonnage reste délicat en pratique [DN12a].

Notre principale contribution en cryptographie à base de réseaux est la conception d'une signature numérique ayant des performances logicielles meilleures que celles construites à partir de la cryptographie « classique » comme RSA et ECDSA, tout en conservant des tailles de clés et signatures comparables aux signatures basées sur RSA, et pouvant être implémentée sur des environnements contraints (par exemple sur carte à puce).

Notre signature numérique s'inscrit dans la lignée de travaux de Lyubashevsky [Lyu08, Lyu09, Lyu12] et est basée sur le paradigme de Fiat-Shamir, dans lequel un système d'authentification en trois passes (un engagement, un défi et une réponse) est transformé en une signature en liant le message et l'engagement par un oracle aléatoire (qui retourne alors le défi). Dans les constructions à base de théorie des nombres (voir par exemple [Sch89] ou [BP02]) l'engagement porte sur une valeur *aléatoire* y qui sera utilisée lors du calcul de la réponse $z = y + \imath(s)$ pour cacher toute information provenant du secret s lors de l'ajout d'une fonction $\imath(s)$ d'icelui. En cryptographie à base de réseaux cependant, les problèmes sous-jacents utilisent des valeurs « petites » ou bornées. Lyubashevsky utilise alors une méthode de rejet [vN51] qui permet d'échantillonner selon une distribution de probabilité f étant donnée une source qui échantillonne selon une distribution g, en rejetant un échantillon x avec une probabilité $f(x)/(M \cdot g(x))$ où M est un réel tel que

$$M \cdot g(x) \geqslant f(x), \qquad \text{pour tout } x. \tag{1.1}$$

Ainsi si g est la distribution de probabilité donnée par $g(x) = \Pr[x = y + \imath(s)]$ (où y et s sont choisis aléatoirement et indépendamment selon leurs distributions respectives), il suffit de choisir une fonction f indépendante de s et d'utiliser la méthode précédente. On obtient alors que la probabilité de ne pas rejeter la valeur est $1/M$ et que les valeurs de sortie sont distribuées selon f (et ne dépendent donc pas de s). Dans le dernier schéma de signature de Lyubashevsky, la fonction f est une Gaussienne discrète (donc bornée avec probabilité écrasante) et la fonction g une Gaussienne discrète centrée en une valeur $\imath(s) = s \cdot c$. Comme on peut le constater sur la Figure 1.1a, le réel M doit être choisi suffisamment grand pour que la condition de l'équation (1.1) soit vérifiée pour tout x excepté pour un nombre négligeable d'entre eux. En particulier la queue de la Gaussienne



(a) Dans le schéma original [Lyu12]

(b) Dans notre schéma

FIGURE 1.1 – Modification du rejet grâce à une distribution Gaussienne bimodale.

« rouge » (*i.e.* de f) sur la projection selon l'axe $s \cdot c$ doit être sous la queue de la Gaussienne « bleue » (*i.e.* de g). Comme une valeur est acceptée avec probabilité $1/M$, on souhaite choisir M le plus petit possible. Dans notre schéma de signature, nous modifions la fonction g en une Gaussienne bimodale centrée en les valeurs $s \cdot c$ et $-s \cdot c$. Cette nouvelle fonction g est plus proche de la fonction de sortie f, et permet ainsi de sélectionner un M plus petit comme on peut le constater sur la Figure 1.1b. Ensuite nous optimisons notre schéma de signature en nous basant sur des réseaux de type NTRU [HPS98, HHGP+03], en utilisant des algorithmes de compression et des astuces algorithmiques pour éviter des calculs nécessitant une grande précision. Nous obtenons finalement des signatures de 5000 bits, pour 128 bits de sécurité heuristique, conduisant à des implémentations logicielles particulièrement efficaces. Notre schéma de signature a aussi été conçu de manière à pouvoir être implémenté efficacement dans des environnements contraints, notamment en concevant une série d'algorithmes simples permettant d'échantillonner selon une Gaussienne discrète sur les entiers en utilisant un nombre logarithmique d'éléments pré-calculés.

Donnons maintenant quelques perspectives sur la cryptographie à base de réseaux. Cette dernière a connu un réel engouement suite aux travaux de Regev [Reg09] qui a introduit le problème LWE, « Learning with Errors », qui se réduit aux pires instances de problèmes algorithmiques sur les réseaux. LWE est très versatile : sa simplicité et sa structure mathématique forte permettent de le décliner de manière originale pour donner lieu à des schémas innovants comme le chiffrement complètement homomorphe [Gen09, BGV12]. Il est courant de dire que toutes les primitives cryptographiques peuvent être instanciées avec des réseaux. Malheureusement, cette versatilité semble être compensée par le caractère peu pratique des schémas obtenus. De plus, cette cryptographie repose sur des « bruits » : il est alors nécessaire de connaitre le protocole utilisé afin de choisir des paramètres en assurant la justesse et la sécurité. Par exemple, Ducas souligne dans sa thèse de doctorat [Duc13] qu'instancier un schéma de chiffrement basé sur l'identité hiérarchique à base de réseaux nécessite de prendre en compte le nombre de niveaux hiérarchiques, contrairement à son analogue basé sur les couplages. Le fait que l'on ne puisse pas utiliser un schéma comme une « boite noire » est aussi illustré par notre signature numérique. C'est en plongeant dans les détails du schéma qu'on a pu proposer des améliorations spécifiques qui ont permis de sélectionner des paramètres et de concevoir une alternative prometteuse aux signatures numériques basées sur la cryptographie classique.

La cryptographie à base de réseaux nécessite de manipuler de nombreux paramètres intimement liés et dont toute modification, même infime, est susceptible d'affecter de manière significative la difficulté des meilleures attaques. Assurer une efficacité optimale tout en maintenant un niveau de sécurité donné semble donc être un problème d'optimisation délicat. Nous pensons qu'une automatisation de la sélection des paramètres serait non seulement utile, mais pourrait se révéler fondamentale dans l'éventualité d'une adoption plus large de cette cryptographie. Par ailleurs, la praticité de notre schéma de signature (ou d'autres récents schémas du domaine) n'est atteinte que grâce à l'utilisation de réseaux idéaux (*i.e.* provenant d'idéaux d'anneaux). Or les problèmes algorithmiques sur ces réseaux structurés n'ont pas été autant étudiés que sur leurs homologues aléatoires ; et s'il n'existe pas de méthode connue pour exploiter cette structure sous-jacente, il

n'est pas inenvisageable que cet état de fait change dans le futur [Ber]. Davantage de cryptanalyse apparait donc nécessaire pour donner confiance en la sécurité de ces schémas. Nous pensons que le déséquilibre actuel existant entre la cryptographie et la cryptanalyse sur les réseaux, la première étant d'avantage représentée dans les travaux récents, peut être dû au manque de paramètres concrets (c'est-à-dire non asymptotiques) proposés. Nous espérons que les instanciations de notre schéma de signature, ciblant des niveaux de sécurité de 128, 160 et 192 bits, provoqueront un regain d'intérêt de la communauté envers la cryptanalyse de cette cryptographie.

La cryptographie à base de réseaux est certainement un champ de recherche dont la versatilité ne cesse d'étonner [GGH+13b], et auquel les premières implémentations très efficaces [GLP12, PG12, PG13, PG14, OPG14] promettent un avenir radieux.

1.2 Contributions au chiffrement complètement homomorphe

La seconde partie de cette thèse est consacrée à la présentation de contributions portant sur le chiffrement complètement homomorphe. Nous avons principalement travaillé à rendre cette primitive, parfois considérée comme le *Saint Graal* de la cryptographie [Mic10], plus efficace et utilisable en pratique pour certaines applications.

Ce chiffrement a été prouvé possible en 2009 par Craig Gentry [Gen09] après être resté pendant trois décennies un problème ouvert [RAD78]. Ce chiffrement permet de manipuler à souhait et publiquement des données chiffrées (donc sans connaitre les données en clair). Cela permet par exemple de sous-traiter des calculs au *Nuage*[2], *i.e.* à des serveurs distants, sans qu'il n'acquière d'information sur les données traitées, qui restent ainsi confidentielles. En particulier il devient possible de réaliser des fonctionnalités complexes, comme faire des recherches croisées entre les données (chiffrées) et une base de donnée publique (*e.g.* une requête d'un moteur de recherche) tout en ne connaissant rien des données elles-mêmes. La conception et l'implémentation de schémas de chiffrement complètement homomorphe (FHE) est un domaine en pleine effervescence.

Parmi les schémas de FHE proposés, nous nous focaliserons sur le schéma DGHV de van Dijk, Gentry, Halevi et Vaikuntanathan [vDGHV10], dont la sécurité repose sur le problème du diviseur commun approché [HG01].

1.2.1 Chiffrement par lots complètement homomorphe sur les entiers [CCK+13]

Nos contributions à ce champ de recherche consistent en une amélioration importante du schéma de van Dijk *et al.*, déjà revisité par Coron, Mandal, Naccache et Tibouchi [CMNT11, CNT12], en permettant à chaque chiffré de contenir plusieurs bits de données, *i.e.* de le transformer en un chiffrement *par lots* et d'effectuer des opérations en parallèle sur ces bits (sur le principe du SIMD). Une telle fonctionnalité avait été décrite pour des schémas de FHE basés sur LWE dans [BGV12, BGH13] mais reposait sur la forte structure mathématique contenue dans les réseaux idéaux. Pour cela, nous avons introduit une variante décisionnelle du problème du diviseur commun approché dans laquelle il s'agit de distinguer si un élément est uniformément distribué modulo $N = p \cdot q$ (où p et q sont secrets et n'ont pas de petits facteurs, et N est le module public), ou s'il est de la forme[3] $\mathsf{CRT}_{p,q}(r, q')$, où q' est uniforme modulo q et r est « petit ». Cette variante est prouvée équivalente à la variante calculatoire (dans laquelle il s'agit de retrouver p depuis des échantillons de la forme $\mathsf{CRT}_{p,q}(r, q')$) dans [CLT14a].

Nous avons conçu un schéma FHE dans lequel un vecteur de bits $\mathbf{m} = (m_1, \ldots, m_n)$ est chiffré en

$$c = \mathsf{CRT}_{p_1,\ldots,p_n,q}(2r_1 + m_1, \ldots, 2r_n + m_n, q')\,,$$

où les r_i sont « petits » et q' est uniforme modulo q, le module public étant $N = p_1 \times \cdots \times p_n \times q$. Armés du problème décisionnel précédent, nous pouvons montrer que cette variante avec plusieurs nombres p_i reste sémantiquement sure, et nous décrivons comment permuter de manière publique les éléments du vecteur $\mathbf{m}$ lors de la procédure de rafraichissement du chiffré. Cette modification nous a permis d'évaluer de façon homomorphe un circuit non trivial (AES) et de traiter plusieurs

[2] Le « cloud » en anglais.

[3] Pour des entiers a, b, p et q, on définit $u = \mathsf{CRT}_{p,q}(a, b)$ comme le plus petit entier positif tel que $u \equiv a \bmod p$ et $u \equiv b \bmod q$.

blocs de données en parallèle. Ainsi, pour 72 bits de sécurité, une évaluation complète du circuit sur 528 blocs de 128 bits prend 113 heures, ce qui donne un temps relatif de moins de 13 minutes de calcul par bloc de donnée.

1.2.2 Minimisation du nombre de bootstrappings dans un circuit homomorphe [LP13]

Malheureusement, les bruits r_i dans le schéma précédent voient leur taille augmenter exponentiellement avec la profondeur multiplicative du circuit. Ainsi, il est nécessaire d'appliquer très régulièrement une procédure publique de rafraichissement sur le chiffré, appelée *bootstrapping*, très couteuse. Que ce soit pour un bit de donnée dans [CNT12] ou n bits de données dans le schéma sus-mentionné, cette procédure prend plusieurs minutes sur des processeurs actuels (à comparer aux millisecondes que prennent les autres opérations). Il convient donc de minimiser le nombre de ces procédures tout en assurant l'intégrité des données au fur et à mesure de l'évaluation homomorphe. Faces à ce problème et soumis à la contrainte précédente, nous avons proposé un modèle théorique du comportement du bruit dans les schémas homomorphes, ainsi qu'une méthode heuristique qui permet de déterminer pour chaque circuit les moments adéquats pour appliquer la procédure de rafraichissement. Notre méthode consiste à construire une formule booléenne pour chaque circuit, et à chercher une implication logique de taille minimale. En général ce problème est prouvé être NP-complet [GHM05] ; nous proposons ainsi une méthode de résolution heuristique qui nous permet d'obtenir des résultats nouveaux, notamment un gain de performance de 88% pour l'évaluation homomorphe de l'AES décrite dans [CCK+13].

1.2.3 Chiffrement complètement homomorphe sur les entiers à module invariant [CLT14a]

Une seconde approche pour pallier l'augmentation exponentielle de la taille du bruit dans [CCK+13] consiste à modifier notre schéma pour le rendre à niveaux, c'est-à-dire tel que l'augmentation précédente n'est plus que linéaire en la profondeur multiplicative du circuit. En premier lieu conçue pour les schémas FHE basés sur LWE [BV11b, BGV12], la technique de commutation de modules a été adaptée à DGHV par Coron, Naccache et Tibouchi dans [CNT12] et repose sur un nouveau problème décisionnel. Elle consiste à convertir un chiffré modulo N en un chiffré modulo un N' plus petit, le bruit étant réduit automatiquement d'un facteur N/N'. Un choix judicieux de chaine de modules permet alors d'obtenir un schéma à niveaux. Cependant pour un circuit ayant une profondeur multiplicative de L, cette technique nécessite de travailler avec une clé publique environ L fois plus importante que dans le schéma classique. Ceci explique pourquoi Gentry, Halevi et Smart ont dû utiliser un serveur avec 256 Go de mémoire vive pour leur évaluation homomorphe de l'AES dans [GHS12c].

À Crypto 2012, Brakerski a introduit une nouvelle technique permettant d'obtenir un schéma à niveaux pour les schémas FHE basés sur LWE [Bra12]. Similaire à la commutation de modules, excepté que le même module est utilisé tout au long de l'évaluation homomorphe, elle permet d'obtenir un schéma FHE à niveaux dit *à module invariant*. Elle est basée sur des chiffrés $\mathbf{c}$ (où $\mathbf{s}$ est la clé secrète) tels que $\langle \mathbf{c}, \mathbf{s} \rangle = \lfloor N/2 \rfloor \cdot m + e \bmod N$ avec e « petit », au lieu de $\langle \mathbf{c}, \mathbf{s} \rangle = m + 2e \bmod N$ dans le schéma initial de Regev [Reg09] ; autrement dit le message est déplacé du bit de poids faible au bit de poids fort modulo N.

Nous avons adapté cette technique à la version améliorée de DGHV de [CNT12] et à DGHV par lots introduit dans [CCK+13]. Un vecteur de bits $\mathbf{m} = (m_1, \ldots, m_n)$ ($n \geqslant 1$) est chiffré en

$$c = \mathsf{CRT}_{p_1^2, \ldots, p_n^2, q}(r_1 + \lfloor p_1/2 \rfloor \cdot m_1, \ldots, r_n + \lfloor p_n/2 \rfloor \cdot m_n, q') \,,$$

où les r_i sont « petits » et q' est uniforme modulo q, le module public étant $N = p_1^2 \times \cdots \times p_n^2 \times q$. Ainsi, le bit de message m_i a été déplacé du bit de poids faible au bit de poids fort de $c \bmod p_i$. Pour rendre ce schéma homomorphe, nous avons changé les p_i en p_i^2, et avons proposé une méthode publique de conversion qui permet, après une multiplication modulo N, de transformer l'entier obtenu en un chiffré valide (*cf.* Figure 1.2). Nous obtenons alors un schéma FHE à niveaux, dont la sécurité repose toujours sur le problème du diviseur commun approché.

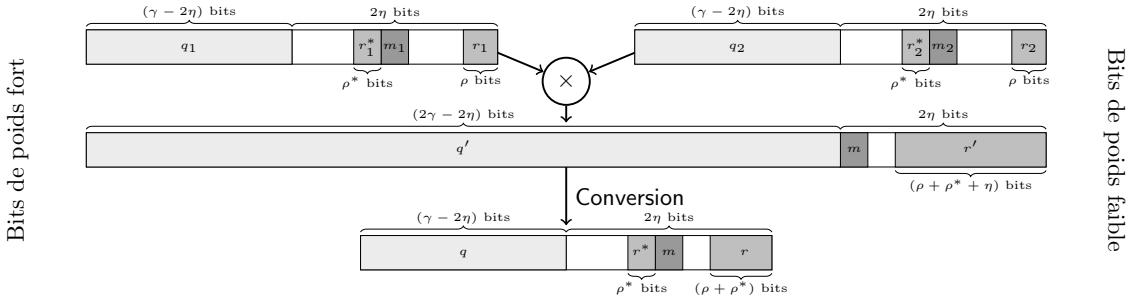


Figure 1.2 – Technique du module invariant pour le schéma DGHV.

Forts de cette nouvelle technique, nous avons à nouveau évalué l'AES de façon homomorphe. Pour 80 bits de sécurité, une évaluation complète du circuit sur 1875 blocs de 128 bits nécessite 102 heures, ce qui donne un temps relatif de 3 minutes de calcul par bloc de données (comparables aux 5 minutes par bloc de [GHS12c]).

1.2.4 Conclusion et perspectives

L'état de l'art en cryptographie homomorphe a significativement changé ces cinq dernières années. En 2009, nous disposions d'un schéma uniquement théorique, dont la sécurité reposait sur des hypothèses fortes ; sa première instanciation décrite en 2011 avait malheureusement une efficacité peu satisfaisante [GH11b]. En 2014, nous disposons de plusieurs grandes familles de FHE (au moins 4), basées sur des problèmes plus classiques et dont la plupart possèdent des implémentations prometteuses. En particulier, des évaluations homomorphes de différents circuits (AES, Simon, Prince) s'exécutent en quelques minutes sur des ordinateurs actuels. Bien que ces performances ne soient pas encore très satisfaisantes [Ber], les évaluations de circuits ayant une faible profondeur multiplicative deviennent très efficaces [NLV11, BLLN13, LN14a]. Il devient donc possible de réaliser des calculs et tests statistiques ou des algorithmes d'apprentissage automatique simples [GLN12] en ne manipulant que des données chiffrées. En particulier, les premiers prototypes utilisant du chiffrement homomorphe devraient voir le jour très prochainement pour des applications sur des données médicales, biométriques ou de géolocalisation.[4] Nous prévoyons dans la suite de notre recherche d'investiguer ce qu'il est possible de réaliser grâce aux schémas de chiffrement homomorphe existants, de trouver des applications qui n'exploiteraient pas leurs faiblesses mais leurs forces, ou à l'inverse de concevoir de nouvelles applications pratiques qui sont adaptées aux contraintes actuelles des schémas FHE. Comme pour la cryptographie à base de réseaux, il semble qu'une automatisation de la sélection des paramètres, ou de la construction de circuits (comme nous l'avons commencé dans [LP13]), va devenir nécessaire pour obtenir des primitives efficaces et sures.

1.3 Contributions aux applications multilinéaires cryptographiques [CLT13b]

Les applications multilinéaires cryptographiques, généralisant les applications bilinéaires [Jou00, SOK00, BF01], ont été considérées dès 2003 par Boneh et Silverberg [BS03] et sont présupposées avoir d'importantes conséquences en cryptographie. Encore aujourd'hui, construire une telle généralisation reste un problème ouvert. En 2013 cependant, Garg, Gentry et Halevi ont proposé une approximation d'applications multilinéaires (appelée système d'encodages gradués) qui certes diffère de la primitive proposée par Boneh et Silverberg mais permet toujours, par exemple, de faire un échange de clé non interactif entre N utilisateurs pour tout N (similairement, l'introduction des couplages permettait alors de faire une telle mise en accord de clés entre $N = 3$ utilisateurs [Jou00]). À contre-courant par rapport à la cryptographie à clé publique moderne, cette nouvelle primitive n'admet qu'une sécurité heuristique (non prouvée). Uniquement théorique, elle se base sur des idées empruntées

[4] Ceci est d'autant plus probable que le dernier appel à projet (orienté vers les industries) de la commission européenne [H20], baptisé *Horizon 2020*, mentionne explicitement le chiffrement homomorphe comme un élément clé de sécurité à mettre en œuvre d'ici quelques années.

aux schémas de chiffrement homomorphes sur les réseaux (notamment [Gen09, LTV12]) et son implémentation parait difficile[5].

Notre principale contribution dans ce domaine est la conception d'un second schéma qui approxime les applications multilinéaires selon le même modèle que [GGH13a], mais pouvant être vu comme émanant des schémas homomorphes DGHV sur lesquels nous avons travaillé tout au long de notre thèse. Comme pour nos autres résultats, nous décrivons également une implémentation « preuve de concept » qui permet de réaliser une mise en accord de clé non interactive entre 7 participants en moins de 40 secondes, avec des paramètres publics de 2.6 Go.

Les encodages dans nos applications multilinéaires sont associés à un niveau, niveau qui n'augmente pas lorsqu'on ajoute les deux encodages (ce qui ajoute les valeurs encodées de façon homomorphe), mais qui augmente en la somme des niveaux des encodages sources lors d'une multiplication (qui multiplie les valeurs encodées de façon homomorphe). En particulier, un vecteur $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ est encodé au niveau i en

$$c = \mathsf{CRT}_{p_1,\ldots,p_n}(g_1 \cdot r_1 + m_1, \ldots, g_n \cdot r_n + m_n)/z^i \bmod N\,,$$

où les r_i sont « petits », z est un masque multiplicatif secret et le module public est $N = p_1 \times \cdots \times p_n$. L'évaluation de l'application multilinéaire consiste à effectuer des additions et multiplications jusqu'au niveau maximal κ (selon le protocole sous-jacent), et ensuite d'extraire d'un encodage au niveau κ (qui contient du bruit) une valeur indépendante du-dit bruit. Ceci est rendu possible en ajoutant dans les paramètres publics une clé de déchiffrement partielle p_{zt} telle que les bits de poids fort de $\omega = c \cdot p_{zt} \bmod N$ ne dépendent que de

$$\sum_{i=1}^{n} \left(h_i \cdot m_i \cdot g_i^{-1} \bmod p_1\right) \cdot \prod_{j \neq i} p_j\,,$$

où les h_i sont des valeurs secrètes de p_{zt}. En particulier, deux encodages du même vecteur vont être tels que les bits de poids fort des ω correspondants sont les mêmes.

Suite à une analyse de sécurité et des optimisations heuristiques, nous avons instancié notre schéma pour réaliser un échange de clé non interactif entre 7 participants, et ainsi prouvé qu'un tel échange (le premier de son genre) pouvait se réaliser en moins de 40 secondes sur un ordinateur usuel.

Suite à la découverte majeure des applications multilinéaires approximatives, une multitude d'applications basées sur celles-ci ont vu le jour. Particulièrement, Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH+13b] ont décrit une nouvelle primitive, objet de mythe en cryptographie : l'obfuscation indistinguable. Schématiquement, cette obfuscation notée iO permet, depuis deux circuits C_0 et C_1 ayant la même fonctionnalité et une taille similaire, de produire un nouveau circuit $iO(C_b)$ d'un des deux circuits sans que l'on puisse déterminer lequel, *i.e.* sans que l'on puisse déterminer b. Elle a été qualifiée de « la meilleure obfuscation [cryptographique] possible » [GR07] puisqu'elle cache autant d'information sur le circuit initial que possible. Assez étonnement, elle nous apparait très semblable à la notion de cryptographie en boite blanche sur laquelle nous avons aussi travaillé pendant notre thèse (*cf.* Section 1.4.2), et nous souhaiterions étudier les relations entre ces deux notions dans le futur. Cette nouvelle primitive a elle aussi eu des conséquences significatives en cryptographie théorique. En pratique cependant, elle reste plutôt inefficace : dans [Cor13], Coron a estimé qu'obfusquer l'AES avec notre implémentation « preuve de concept » – qui est la seule disponible à ce jour – prendrait $2 \cdot 10^{62}$ années ! Le problème ouvert fondamental est ainsi de construire des applications multilinéaires qui peuvent gérer un très grand nombre de niveaux. Leur similitude en essence avec le chiffrement homomorphe laisse espérer que toute avancée dans un des domaines aura des retombées immédiates dans le second. En nous basant sur les avancées obtenues sur les schémas FHE, nous prédisons que dans les 5 années à venir, des applications multilinéaires de 20 à 30 niveaux seront possibles en quelques millisecondes (*i.e.* trois ordres de magnitude plus rapide que l'implémentation actuelle), et encourageons ainsi la communauté cryptographique à rechercher des applications utilisant un nombre « petit » de niveaux (à l'inverse de l'obfuscation qui en nécessite des millions).

[5] En instanciant naïvement les paramètres asymptotiques suggérés, nous obtenons que les paramètres publics doivent faire de l'ordre de 3400 To pour un échange de clé non interactif entre 7 utilisateurs.

1.4 Autres travaux

Nous avons par ailleurs mené quelques travaux qui ne sont pas détaillés dans cette thèse ; soit parce qu'ils ont été finalisés concurremment [LN14a] ou qu'ils ne relèvent pas de l'un des trois axes sus-cités [LRM+13, DLPR13a].

1.4.1 Comparaison de chiffrements complètement homomorphes basés sur les réseaux [LN14a]

Le chiffrement homomorphe est considéré comme un des éléments les plus prometteurs pour assurer la sécurité du Nuage, tout en permettant à ce dernier de proposer une expérience riche aux utilisateurs. Malheureusement, tous les schémas existants entrainent une expansion de chiffré (*i.e.* la taille du chiffré par rapport à la taille du message initial) tellement importante qu'elle rend inenvisageable l'envoi des données chiffrées avec un schéma homomorphe. Pour pallier cela, des solutions hybrides ont été proposées dans lesquelles les données sont transmises chiffrées sans expansion de chiffré (*e.g.* avec un schéma de chiffrement par blocs) puis déchiffrées de façon homomorphe avant d'être manipulées. Réaliser de telles évaluations homomorphes est un sujet d'actualité auquel nous avons participé en évaluant AES dans [CCK+13, CLT14a].

Notre contribution dans cet article est plurale. Tout d'abord, nous proposons d'évaluer Simon [BSS+13] plutôt que l'AES, c'est-à-dire un schéma de chiffrement par blocs léger[6] conçu pour être efficace sur architecture matérielle. En effet, du fait des contraintes actuelles des schémas de FHE, cette approche est susceptible d'avoir un fort impact sur l'efficacité des évaluations. Il s'avèrera plus tard que considérer Prince [BCG+12] pourrait être un choix encore plus judicieux [DSES14]. D'autre part, nous comparons deux schémas FHE à base de réseaux et à module invariant [FV12, BLLN13], en théorie et en pratique, afin d'en évaluer les forces et faiblesses respectives. Nos choix de paramètres reposent sur une amélioration de l'approche de van de Pol et Smart [vdPS13] et de la version complète de l'article de BKZ-2.0 [CN13].

Il apparait que les deux schémas considérés sont non seulement plus efficaces que les schémas DGHV sur lesquels nous avons travaillé pendant notre thèse, mais offrent de très belles performances sur des circuits de faible profondeur.

1.4.2 Cryptographie en boite blanche [LRM+13, DLPR13a]

Le modèle d'attaque en boite blanche a été introduite en 2002 par Chow, Eisen, Johnson and van Oorschot [CEJvO02b, CEJvO02a] comme le pire modèle d'attaque possible. En effet, celui-ci considère un attaquant qui a une connaissance totale de l'implémentation de l'algorithme et contrôle à souhait l'environnement d'exécution (plus représentatif des menaces réelles actuelles). L'idée de la cryptographie en boite blanche est de proposer des implémentations de schémas telles que la clé reste totalement secrète sous le modèle d'attaque sus-cité. Les travaux de Chow *et al.* ont provoqué une vague de d'implémentations candidates pour le DES [CEJvO02b, LN05, WP05] et l'AES [CEJvO02a, BCD06, XL09, Kar10]. Malheureusement, celles-ci ont toutes été suivies d'attaques très efficaces [JBF02, BGEC04, GMQ07, WMGP07, MGH08, MWP10, MRP12, LRM+13].

En particulier, dans [LRM+13], nous décrivons une attaque en 2^{22} contre la dernière implémentation en boite blanche de l'AES supposée sure [Kar10]. Plus qu'une simple cryptanalyse, nous mettons en évidence que l'approche heuristique consistant à transformer des schémas de chiffrement en réseaux de tables de correspondance admet des faiblesses inhérentes. Il est possible que cette succession de candidats heuristiques et d'attaques dévastatrices puisse venir d'un manque de clarté sur ce qui est réellement attendu par la cryptographie en boite blanche. Poussés par cette question, nous traduisons dans [DLPR13a] les intuitions folkloriques égrainées dans différents articles en proposant des notions de sécurité concrètes qu'un compilateur en boite blanche peut atteindre. Nous présentons aussi des constructions qui atteignent certaines de ces notions en s'inspirant de primitives à clé publique. Nos résultats ouvrent ainsi de nouvelles perspectives sur la conception de programmes résistants aux attaques en boite blanche.

[6] En anglais, on qualifie ces schémas de « lightweight ».

1.5 Liste de publications

Sont listées ci-dessous, par ordre chronologique croissant, toutes nos publications dans des conférences ou groupes de travaux internationaux. Lorsque les versions complètes et/ou les implémentations « preuves de concept » sont disponibles en ligne, nous donnons également les références correspondantes.

[JL11] **Traitor Tracing Schemes for Protected Software Implementations.**
M. Joye, T. Lepoint. (ACM-DRM 2011)

[JL12] **Partial Key Exposure on RSA with Private Exponents Larger than N.**
M. Joye, T. Lepoint. (ISPEC 2012)

[LP13] **On the Minimal Number of Bootstrappings in Homomorphic Circuits.**
T. Lepoint, P. Paillier. (WAHC 2013)

[CCK+13] **Batch Fully Homomorphic Encryption over the Integers.**
J.H. Cheon, J.-S. Coron, J. Kim, M.S. Lee, T. Lepoint, M. Tibouchi, A. Yun. (EUROCRYPT 2013)
La version complète de l'article est disponible en ligne [CLT13a].

[LRM+13] **Two Attacks on a White-Box AES Implementation.**
T. Lepoint, M. Rivain, Y. De Mulder, P. Roelse, B. Preneel. (SAC 2013)
La version complète de l'article est disponible en ligne [LR13].

[DLPR13b] **White-Box Security Notions for Symmetric Encryption Schemes.**
C. Delerablée, T. Lepoint, P. Paillier, M. Rivain. (SAC 2013)
La version complète de l'article est disponible en ligne [DLPR13a].

[DDLL13a] **Lattice Signatures and Bimodal Gaussians.**
L. Ducas, A. Durmus, T. Lepoint, V. Lyubashevsky. (CRYPTO 2013)
La version complète de l'article est disponible en ligne [DDLL13b].
Implémentation « preuve de concept » par L. Ducas et T. Lepoint [DL13].

[CLT13b] **Practical Multilinear Maps over the Integers.**
J.-S. Coron, T. Lepoint, M. Tibouchi. (CRYPTO 2013)
La version complète de l'article est disponible en ligne [CLT13c].
Implémentation « preuve de concept » par T. Lepoint [Lep13].

[CLT14a] **Scale-Invariant Fully Homomorphic Encryption over the Integers.**
J.-S. Coron, T. Lepoint, M. Tibouchi. (PKC 2014)
La version complète de l'article est disponible en ligne [CLT14b].

[LN14a] **A Comparison of the Homomorphic Encryption Schemes FV and YASHE.**
T. Lepoint, M. Naehrig. (AFRICACRYPT 2014)
La version complète de l'article est disponible en ligne [LN14b].
Implémentation « preuve de concept » par T. Lepoint [Lep14].

Chapter 2

Introduction

2.1 Introduction to Cryptology

The *Oxford Dictionary of English* proposes a simple, yet incomplete, definition of cryptology as "the study of codes, or the art of writing and solving them". The roots of this definition are to be found in History: the invention of cryptology comes from the problem of *secret* communications of diplomatic and military information. The basic idea is to apply a "complicated" transformation to the information to be protected. On one side of cryptology, users utilize secret codes, while on the other side, adversaries attempt to break through the secrecy of the messages to recover the hidden information. One of the oldest and simplest cryptologic technique, *Caesar's Cipher*, consists in replacing each letter of your message by the letter three positions down the alphabet (looping back at the end). In the ninth century, the Arab mathematician Al-Kindi showed that such a *substitution cipher*, that is the technique where each letter is replaced with another letter consistently, is vulnerable to a frequency analysis technique. By comparing the frequency of the letters in the language (*e.g.* in English, the letter e occurs 13 percent of the time and a letter probably begins with "Dear Sir:" [DH76]) with the frequencies of the characters appearing in the ciphered message, one can easily recover the hidden message.

Until the XIXth century, the study of secret codes lacked a precise and consistent theory, and designing or breaking such codes was considered as an art [Bab64, Chapter XVIII]. The construction of "good codes" or deciphering relied on time, patience, ingenuity, inventiveness and novelty. With the introduction of mathematical formalism, the study of secret codes became a science – *cryptology*. This science contains two aspects: *cryptography* that aims at designing new methods to ensure the secrecy of communications, and *cryptanalysis* that aims at discovering flaws in these methods. And even though it was usual to keep these methods secret to make cryptanalysis more complicated, such a secrecy "by obscurity" is recognized to be delusive. In 1883, Auguste Kerckhoffs states the principle that a cryptographic system should use a *public* algorithm, that itself uses a small secret information (that can be transmitted easily): the *key* [Ker83]. As a consequence the cryptographic systems can now be scrutinized by the public, and a system that survives years of serious cryptanalytic attention ends up being more trusted than a secret system that no analyst has reviewed.[1]

In this age of digital information and telecommunications, cryptography is now far from being restricted to the military and diplomatic fields. It has become a *cornerstone* in our daily life. Cryptography is present in our cellphones, our banking cards, our biometric passports, our Internet browsers, and many (often unsuspected) other products, which all require to guarantee security properties on their communications and on their data. Moreover, beyond the *confidentiality* of the

[1] Note that the importance of public scrutiny remains an essential component of today's cryptography. This is illustrated for example by the process of *standardization*, in which a group of recognized researchers consensually selects a set of algorithms that meet some desired requirements. This standardization process includes a competition, in which cryptographic systems are not only proposed but scrutinized for years, in order to gain trust in the finally selected systems. Recent cryptographic competitions include the five-year SHA-3 competition [NIS12] (2007-2012) aiming at developing a new cryptographic hash algorithm, called SHA-3, for standardization. On March 15th 2014, all the submissions of authenticated ciphers to the CAESAR competition [CAE16] were made public for public scrutiny. This evaluation phase will help the committee to choose finalist systems by the end of 2016.

secret information, one should also ensure that these contacts are secure against eavesdropping or injection of illegitimate messages. Thus, the scope of cryptography now includes among other things data *integrity* (*i.e.* the fact that the data has not been modified) and data *authenticity* (*i.e.* the fact that the sender is legitimate). Therefore, the cryptographer aims at designing systems that ensure these security properties, while the cryptanalyst looks at possible flaws that would reveal that these properties are actually not verified.

2.2 Modern Cryptography

In their groundbreaking paper *New directions in Cryptography* [DH76] published in 1976, Whitfield Diffie and Martin E. Hellman introduced the concept of public-key cryptography and bridged cryptography to complexity theory.

Until then, all the cryptographic systems were relying on a common secret shared between the sender and the receiver, *i.e.* were using a symmetric secret key (symmetric because it was the same for both parties). A typical example of a symmetric encryption scheme is a block cipher. Such a cryptosystem is a pair of families $\{E_k\}_{k\in\mathsf{K}}$ and $\{D_k\}_{k\in\mathsf{K}}$ of algorithms representing invertible transformations over blocks of fixed length (*e.g.* 128 bits), inverse of each other, indexed by a symmetric key $k \in \mathsf{K}$. When the sender – conventionally named Alice –, who is sharing a common secret key k with the receiver – Bob –, wants to confidentially send a message m to Bob, she can send the *ciphertext* $c = E_k(m)$ from which Bob can recover m by decrypting c: $m = D_k(c)$. Block ciphers remain fundamental and very useful ingredients of today's cryptography; they are extensively used in nearly all systems using cryptography.[2]

Key Exchange. All symmetric key cryptography (also called *secret key* cryptography) assumes that the two parties exchanging secret messages share a common secret key. Unfortunately, the secure distribution of such a key is a major issue. In [DH76], Diffie and Hellman described a very simple approach to eliminate the need for a secure key distribution channel. (Indeed, sending the key in advance over a secure channel is unrealistic for today's applications.) This key exchange is an efficient solution to the problem of creating a common secret between two participants (that can subsequently be used to encrypt all the communications thanks to a symmetric cipher). Moreover, it is one-round, *i.e.* each participant is allowed to talk once and broadcast some data to the other participant.

The parameters (on which Alice and Bob have obviously to agree) consist of a cyclic group $\mathbb{G}$ (denoted additively) of prime order p, generated by $g \in \mathbb{G}$. Alice (resp. Bob) generates a key pair $(\mathsf{sk}_A, \mathsf{pk}_A) = (x, x \cdot g)$ (resp. $(\mathsf{sk}_B, \mathsf{pk}_B) = (y, y \cdot g)$) where $x, y \in \mathbb{Z}_p$ are random and makes the public key pk_A (resp. pk_B) openly available. When Alice and Bob want to share a secret, they both can compute a shared secret key $(xy) \cdot g = y \cdot (x \cdot g) = x \cdot (y \cdot g)$ with their own secret key sk_A (or sk_B) and the other's public key pk_B (or pk_A). The main idea behind the security of the protocol is that an adversary – called Eve – spying on the insecure channel is able to break the protocol if she can compute $(xy) \cdot g$ from $x \cdot g$ and $y \cdot g$. This problem is called the *Computational Diffie-Hellman problem* (CDH).[3] The most efficient attack known against this problem consists in recovering x from $x \cdot g$ (or y from $y \cdot g$), which is exactly the *discrete logarithm problem* (DL). Now, the best known algorithm to compute the discrete logarithm in the cyclic group $\mathbb{G} = (\mathbb{Z}_p^*, \times)$ (considered in [DH76]) is sub-exponential; these results give confidence on the security of the protocol.[4]

This key exchange was extended to three participants by Joux [Jou00] in 2000, using the discrete logarithm problem on elliptic curves and bilinear maps (*i.e.* bilinear and non-degenerate applications $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p). A theoretical generalization [BS03], assuming the existence of cryptographic multilinear maps, easily extends the

[2] A typical block cipher is the *Advanced Encryption Scheme* (AES) [FIP01], that we will briefly recall in Chapter 10.

[3] The decisional variant of this problem, called the *Decisional Diffie-Hellman problem* (DDH), is such that one has to distinguish the distributions $(a \cdot g, b \cdot g, (ab) \cdot g)$ and $(a \cdot g, b \cdot g, c \cdot g)$ for random $a, b, c \in \mathbb{Z}_p$ where g generates a cyclic group $\mathbb{G}$ of prime order p.

[4] Even though the initial group proposed by Diffie-Hellman is $\mathbb{G} = (\mathbb{Z}_p^*, \times)$, note that the discrete logarithm problem is meaningful in an arbitrary cyclic group. However, the resulting problem is not necessarily hard, *e.g.* the DL problem in $\mathbb{G} = (\mathbb{Z}_p, +)$ is straightforward.

result to N participants. In 2013, Garg, Gentry and Halevi described a candidate approximate multilinear maps scheme [GGH13a] that differs from the generalization of [BS03], but still allows to perform a one-round N-partite Diffie-Hellman key exchange. In this thesis, we design a similar approximate multilinear maps scheme in Part III, and describe a N-partite Diffie-Hellman key exchange in Chapter 12.[5] Of course, without using multilinear maps, there exist key exchange protocols with several rounds that establish a common secret between $N > 2$ users.

Public Key Cryptography. As illustrated by the protocol of above, the paradigm in public key cryptography is that the sender and the receiver do not have to share a common secret. Instead, they both have a secret key and a public key – that is therefore not secret. Public key cryptography is also known as *asymmetric* cryptography; the term "asymmetric" stems from the use of different keys to perform the cryptographic functions instead of the same key as in the conventional secret key cryptography.

For example, assume Alice wants to send a (confidential) message to Bob. She can use his *public key* pk – *e.g.* available from a public repository, or previously sent in the clear by Bob – and apply an encryption algorithm with this public key to obtain a ciphertext $c = \mathsf{Encrypt}(\mathsf{pk}, m)$. Note that the whole point is that pk does not allow to recover m from c. This requires the knowledge of the *secret key* sk that Bob kept secret; he is therefore the only one able to decrypt c to recover $m = \mathsf{Decrypt}(\mathsf{sk}, c)$. The first realization of a public key encryption scheme [RSA78] is due to Rivest, Shamir and Adleman in 1978, and is called nowadays "textbook RSA".

In this scheme, the key generation samples two (large) primes p and q, and defines the public modulus $N = pq$. Next a public exponent $e \in \mathbb{Z}$, coprime with $\phi(N) = (p-1)(q-1)$, is selected (often not randomly). Since e is invertible modulo $\phi(N)$, there exists an integer $d \in \mathbb{Z}_{\phi(N)}$ such that $e \cdot d \equiv 1 \pmod{\phi(n)}$. The public key pk is the pair (N, e) and the secret key sk is the integer d. To encrypt a message $m \in \mathbb{Z}_N^*$, one computes $c = m^e \bmod N$. To decrypt a ciphertext $c \in \mathbb{Z}_N^*$, one computes $m = c^d \bmod N$. (The latter operations are indeed inverse of each other by Euler's theorem.) The key idea behind the hardness to recover sk from pk is that, computing d from (N, e) is essentially equivalent to factorizing N. Indeed, from p and q, we can compute $\phi(N)$ and therefore the inverse of e modulo $\phi(N)$. Reciprocally, from d and e, one can recover a multiple of $\phi(N)$, and then $\phi(N)$ using Miller's algorithm [RSA78]. Finally, from $p + q = N - \phi(N) + 1$ and $pq = N$, one can recover p and q easily. Now, we do not know how to factorize a product N of two large primes efficiently; the most efficient known algorithm, the General Number Field Sieve (GNFS) [LJMP90], is sub-exponential in the size of N. (Factorization is still nowadays one of the most important supposedly hard problems of cryptography.) Another attack against the protocol is to recover m from c without using sk (*i.e.* to compute directly e-th roots modulo N). This problem, called the *RSA problem*, is a priori easier than to factor N (otherwise one could recover d and thus m), but as for today it remains an open problem to use the fact that one could know how to compute e-th roots modulo N to factorize N. Once again, this supposed hardness gives confidence on the security of the protocol.

Diffie and Hellman also observed that public key cryptography allows to digitally produce signatures tied to a message m (therefore allowing to check its integrity).[6] Alice owns a secret signing key sk and a public verification key pk. The signing key is used to construct a signature $\sigma = \mathsf{Sign}(\mathsf{sk}, m)$ of a message m. This signature is publicly verifiable using the public key pk: $\mathsf{Verify}(\mathsf{pk}, \sigma, m) = \mathtt{true}$ if and only if σ was obtained as previously. As previously, the knowledge of pk does not allow to recover sk, nor to issue valid signatures for a message m.

Unfortunately public key cryptography is, in practice, noticeably less efficient than symmetric cryptography. To combine the advantages of both cryptographies, we use a conjunction of thereof. This is what we call a *hybrid cryptosystem*. Such a cryptosystem combines the convenience of a public key cryptosystem with the efficiency of a symmetric key cryptosystem. In particular, we use

[5] The first benchmark of our alternative construction shows that it is possible to realize a 7-partite key exchange in a matter of seconds.

[6] Signatures allow to ensure the integrity and the authenticity of a message by everyone because it only relies on the knowledge of the *public* verification key. Achieving such a feature is not possible using only symmetric cryptography. In secret key cryptography, two participants sharing a secret key k can mutually ensure the authenticity and integrity of their messages using a cryptographic primitive called message authentication code, or MAC. However, they cannot convince a third party which does not possess the key k.

a public key cryptosystem to encrypt a symmetric key, that will be subsequently used to encrypt the data. Therefore, one only has to use public key cryptography to send a small amount of data (roughly the symmetric key) while the data can be decrypted efficiently using *e.g.* a block cipher. Classical examples of hybrid cryptosystems include the `OpenPGP` file format and `PKCS #7` file format, both used by many different systems.[7]

In this thesis, we will design an efficient "lattice-based" (*cf.* Section 2.3) signature scheme in Part I and a public key encryption scheme (with additional features) in Part II.

Heuristic but Proven Security. A public key encryption scheme is a *one-way function*. Indeed, the function which, to a message m, associates the ciphertext $c = \mathsf{Encrypt}(\mathsf{pk}, m)$ has to be efficiently computable (*i.e.* in polynomial time) for the cryptosystem to be of any use. Also, it has to be hard to invert: it must be difficult to publicly recover m from c. Now, the existence of one-way functions implies $\mathsf{P} \neq \mathsf{NP}$, which is probably the most famous open problem in theoretical computer science. Therefore, all of public key cryptography is heuristic; generally we assume that some problems are difficult (*i.e.* impossible to solve in polynomial time) even though we do not know how to prove it. In cryptography, integer factorization and the discrete logarithm problem, previously mentioned as arguments for the security of RSA and the Diffie-Hellman key exchange, are supposed to be hard problems. Various other problems are considered (such as knapsack problems, decoding problems for linear codes or resolution of large polynomial systems), but are often not used in practice. A very fruitful family of problems that allowed to construct numerous cryptographic primitives in the last years is based on lattices. One of the main purposes of this thesis is to study lattice-based cryptography, *i.e.* cryptography based on lattices.

A legitimate question one could ask is: When can we say that a cryptographic primitive is secure (even though its security may only be heuristic)? For example, "textbook RSA" cannot be considered as secure. Indeed, one can recover any message smaller than $N^{1/e}$ by extracting e-th roots over the integers, or can decide whether two ciphertexts correspond to the same plaintext. In order to give confidence in public key cryptography, cryptographer introduced new security notions. To prove the security of a cryptosystem, cryptographers consider attack scenarios in which an adversary is given a *black-box* access to the cryptographic system, namely to the inputs and outputs of its underlying algorithms. Security notions are built on the standard paradigm that the algorithms are known and that computing platforms can be trusted to effectively protect the secrecy of the private key.[8] Once the security notions were clearly defined, it has been possible to create public key schemes with *proofs* that they achieved the latter notions, under the hypothesis that some problem is difficult. In such a proof, the capabilities of the attacker are defined by an attack model and the aim of the proof is to show that an attacker attacking efficiently (*i.e.* in polynomial time) the security notion can be used in order to solve efficiently the underlying (supposedly hard) problem. Now, if the problem is assumed to be hard, it gives confidence that no such efficient adversary can exist.

Security proofs are certainly among the most remarkable achievements of modern cryptography. They provide a strong form of evidence that an application does reach the required security strength (80 bits, 128 bits, 256 bits, etc.). Now widely adopted by certification bodies and standardization organizations, security proofs may also serve just as an eye-opener for security architects. In this thesis, we provide security proofs for the schemes in Parts I and II, while the security of the scheme described in Part III will be mainly heuristic.

[7] Jumping ahead, note that white-box cryptography (see Section 2.4) would also be a promising solution that combines the advantages of public key cryptography (more precisely, that it does not require a shared secret) and secret key cryptography (more precisely, its speed). Indeed, assume that Bob can create a white-box AES implementation $[E_k]$ with an embedded key k. By definition, $[E_k]$ does not allow to recover any information on k beyond what a black box access would reveal. Therefore we automatically transformed a secret key encryption scheme (AES with secret key k) into a public key encryption scheme where $\mathsf{pk} = [E_k]$ and $\mathsf{sk} = k$, except that the decryption operation – performed by Bob – only uses the decryption algorithm of AES with the secret key k, and is thus much more efficient than "classical" public key decryption.

[8] However attacks on *implementations* of cryptographic primitives have become a major threat due to side-channel information leakage such as execution time, power consumption or electromagnetic emanations (see *e.g.* the surveys [Joy09, Roh09]). More generally, the increasing penetration of cryptographic applications onto untrusted platform (the end points being possibly controlled by a malicious party) makes the black-box model too restrictive to guaranty the security of *programs* implementing cryptographic primitives.

All the works presented in this manuscript fall in the scope of public key cryptography. We will use some secret key primitives such as SHA-2 [FIP12] in Chapter 6 and AES [FIP01] in Chapter 10.

2.3 Lattice-Based Cryptography

Asymmetric cryptography usually relies on simple algebraic structures. Groups for which the RSA problem or the discrete logarithm problem are hard suffice to construct the main cryptographic primitives such as asymmetric encryption or signatures. However, richer algebraic structures gradually allowed to design new primitives and protocols. For example during the last decade, bilinear maps (pairings) [Jou00, SOK00, BF01] on adequate groups made possible to construct public key *identity-based* encryption (*e.g.* [BF01]), and simpler and more efficient advanced protocols such as e-voting and e-cash.

An *Euclidean lattice* is a regular arrangement of points in an Euclidean space. Recently in cryptology, lattices have known a renewal of interest. They were implicitly used in cryptography in knapsack-based cryptosystems [Odl90] and in cryptanalysis [NS01].[9] Lattices were (re)discovered as a source of computational hardness for the design of secure cryptographic functions following the seminal works of Ajtai [Ajt96] and Regev [Reg09]. Rapidly expanding, lattice-based cryptography is now considered as the most promising alternative to traditional cryptography [LLS14]. In particular, lattice-based cryptography has been recognized for its many very attractive selling arguments. Not only it unlocked abundant new cryptographic primitives (including powerful tools like fully homomorphic encryption), but it also has strong provable security guarantees, apparent resistance to quantum attacks and high asymptotic efficiency.

Security Arguments. In 1996, the seminal work of Ajtai [Ajt96] proposed an average-case problem, now called the *Short Integer Solution* problem (SIS), and showed that solving (average) instances of this problem is as hard as solving *worst-case* problems defined over lattices. Another breakthrough result is the introduction of the *Learning with Errors* problem (LWE) by Regev in 2005 [Reg09], whose average-case instances are (quantumly) as hard as worst-case instances of lattice problems (see also [BLP+13] for a partial dequantization of this reduction). We defer to Section 3.2.2 for some details on these problems. Therefore, lattice-based cryptographic systems admit, most of the time, security proofs under some specific worst-case problems, as opposed to the (average-case) security proofs *for specific input distributions* present in "classical" asymmetric cryptography.

Another selling argument in favor of lattice-based cryptography is that it is currently not known how to exploit quantum computing to solve standard lattice problems significantly more efficiently than with classical computers. This contrasts with classical hardness problems considered in "classical" asymmetric cryptography, such as integer factorization or the discrete logarithm problem, that can all be solved in polynomial-time using a quantum computer [Sho97].

Functionality Arguments. Lattice-based cryptographic primitives rely on simple and flexible operations. Lattices can be tweaked in an original manner to produce groundbreaking schemes such as fully homomorphic encryption (FHE) [Gen09], cryptographic multilinear maps [GGH13a], attribute-based encryption for all circuits [GVW13] or indistinguishability obfuscation [GGH+13b] (among others). In this age of cloud computing, new cryptographic functionalities are now desired; in particular, computations on encrypted data, confidentiality, integrity and verifiability of client data against an untrusted cloud provider, proof of data possession, and access control. Lattice-based cryptography seems the only current cryptographic mean to bring the malleability required by the new usages, while still providing strong security arguments (see previous paragraph).

A fundamental paradigm shift in asymmetric cryptography is that of functional encryption, which enables fine-grained control of access to encrypted data. In particular, such a scheme allows to decide of a decrypting policy not for a unique user but for a set of users: more precisely the owner of a "master" secret key can release restricted secret keys that reveal a specific function of encrypted data [AGVW13]. Functional encryption *for all circuits* was shown to be theoretically possible in the groundbreaking work of Garg, Grentry, Haveli, Raykova, Sahai and Waters [GGH+13b].

[9] Euclidean lattices also have many applications in computer science and mathematics, including the solution of integer programming problems, sphere packings, Diophantine approximations, and many more.

Such an encryption scheme therefore allows to set up access policies on data in the cloud with a cryptographic security [LLS14].

Fully homomorphic encryption (FHE), the *Holy Grail of cryptography* [Mic10] [sic] – as its existence was subject to debates for three decades, enables one to process any function on encrypted data. Thereby it allows to distribute cryptographic tasks to *untrusted* distributed computing resources (such as a cloud infrastructure). In particular, a remote system can provide complex functionalities, like a database system capable of indexing and searching our data, while knowing *nothing* about the data itself. The first FHE scheme was described by Gentry in 2009 [Gen09], and design and applications of FHE has become a major research subject these past five years.

It is worth noting that all the abovementioned cryptographic primitives can be viewed as instantiations of lattice-based cryptography and are not achievable by more usual means.

Efficiency Arguments. Last but not least, lattice-based cryptography enjoys very low asymptotic complexity, as opposed to classical cryptography. Indeed, encryption schemes relying on integer factorization or the discrete logarithm problem are inherently slow [Ste11]. In particular, operations typically cost $\mathcal{O}(n^{2+\epsilon})$ using fast integer multiplication where n is the bit-size of the key pair, and the best known attacks are sub-exponential (typically $2^{\tilde{\mathcal{O}}(n^{1/3})}$ bit operations[10]) with respect to the key length. For a security parameter $\lambda = n^{1/3}$, this means that encryption and decryption usually cost $\Omega(\lambda^6)$. On the other hand, lattice-based cryptography enjoys quasi-linear encryption and decryption cost $\tilde{\mathcal{O}}(\lambda)$ when using ideal lattices, and can be proved as hard to break as solving a computational problem which is believed to require $2^{\Omega(\lambda)}$ time.

Some Critics on the Selling Arguments. Although lattice-based cryptography is asymptotically efficient, its practical instantiations are not always competitive with "classical" asymmetric cryptography instantiations. Even nowadays, the most efficient lattice-based encryption scheme remains NTRUEncrypt [HPS98] (with impressive encryption and decryption performances) whose security relies on heuristic arguments. Theoretically secure lattice-based cryptography has been steadily developed these past years. Yet, at the beginning of this thesis, provably secure lattice-based schemes were rarely implemented, if at all. Moreover, the final key sizes were at least one order of magnitude larger than for "classical" cryptography. Nowadays, to obtain efficient lattice-based cryptosystems with relatively small key sizes, we work over *ideal* lattices (instead of random lattices) whose hardness is unfortunately slightly less understood. The selected parameters are based on (extensive) cryptanalysis of existing algorithms, and do not verify the requirements for the worst-case to average-case reductions (which was a strong selling argument of lattice-based cryptography). Finally, theoretically secure lattice-based cryptography uses discrete Gaussian sampling, which in general requires to work with floating point numbers [DN12a].

Throughout the thesis we focused on improving the efficiency of lattice-based cryptography and other asymmetric cryptosystems considered impractical. In particular, in Chapter 4 we made possible to sample according to a discrete Gaussian over the integers on constrained devices, in Chapters 5 and 6, we designed and implemented an efficient lattice-based signature scheme with small key and signature sizes (as in "classical" cryptography). In Part II we improved the efficiency of FHE schemes of several orders of magnitude and in Part III we described the first implementation of cryptographic multilinear maps, and obtained arguably practical results.

2.4 List of Publications

In this section, we provide an exhaustive list of publications (with their full versions) and implementations to this date, cosigned by ourselves.

Internship Articles. Before this thesis, two publications [JL11, JL12] were produced during an internship at Technicolor, under the guidance of Marc Joye. We provide their abstracts for com-

[10] The notation $\tilde{\mathcal{O}}(\cdot)$ hides poly-logarithmic factors, *i.e.* $f(n) = \tilde{\mathcal{O}}(g(n)) = \mathcal{O}(g(n)\log^c(n))$ for some fixed constant c.

pleteness, but will not discuss them further in this manuscript.

[JL11] **Traitor Tracing Schemes for Protected Software Implementations.**
M. Joye, T. Lepoint. (ACM-DRM 2011)
This paper considers the problem of converting an encryption scheme into a scheme in which there is one encryption process but several decryption processes. Each decryption process is made available as a protected software implementation (decoder). So, when some digital content is encrypted, a legitimate user can recover the content in clear using its own private software implementation. Moreover, it is possible to trace a decoder in a black-box fashion in case it is suspected to be an illegal copy. Our conversions assume software tamper-resistance.

[JL12] **Partial Key Exposure on RSA with Private Exponents Larger than N.**
M. Joye, T. Lepoint. (ISPEC 2012)
In 1998, Boneh, Durfee and Frankel described several attacks against RSA enabling an attacker given a fraction of the bits of the private exponent d to recover all of d. These attacks were later improved and extended in various ways. They however always consider that the private exponent d is smaller than the RSA modulus N. When it comes to implementation, d can be enlarged to a value larger than N so as to improve the performance (by lowering its Hamming weight) or to increase the security (by preventing certain side-channel attacks). This paper studies this extended setting and quantifies the number of bits of d required to mount practical partial key exposure attacks. Both the cases of known most significant bits (MSBs) and least significant bits (LSBs) are analyzed. Our results are based on Coppersmith's heuristic methods and validated by practical experiments run through the SAGE computer-algebra system.

Lattice-Based Cryptography. In collaboration with colleagues from École Normale Supérieure, we designed and implemented the most efficient – up to this date – lattice-based signature scheme. This scheme improves over *all* the others lattice-based signature schemes and is faster than `OpenSSL` implementations of RSA and ECDSA. It therefore appears as a promising post-quantum signature scheme.

An article was published at CRYPTO 2013 [CG13a], and the associated proof-of-concept implementation is openly available. Full details are provided in Part I of this manuscript.

[DDLL13a] **Lattice Signatures and Bimodal Gaussians.**
L. Ducas, A. Durmus, T. Lepoint, V. Lyubashevsky. (CRYPTO 2013)
Full version available at [DDLL13b].

[DL13] **A Proof-of-concept Implementation of BLISS.**
L. Ducas, T. Lepoint.

Fully Homomorphic Encryption. In collaboration with Pascal Paillier (from CryptoExperts), Jean-Sébastien Coron (from University of Luxembourg), Mehdi Tibouchi (from NTT Secure Platform Laboratories) and Michael Naehrig (from Microsoft Research), we improved upon and implemented three fully homomorphic encryption schemes. In particular, homomorphic evaluations of block ciphers were successfully performed, and are either competitive with existing results (with different schemes) or even faster (when changing the underlying block cipher).

An article cosigned with Pascal Paillier was published at the first Workshop on Applied Homomorphic Cryptography [ABS13], a new workshop that aims to bring together researchers, practitioners and industry to present, discuss and share the latest progress in encrypted computing. Full details on this work are available in Chapter 9 of this manuscript.

Two articles cosigned with Jean-Sébastien Coron and Mehdi Tibouchi were published (the first article was merged with an independent work of J.H. Cheon, J. Kim, M.S. Lee, and A. Yun) respectively at EUROCRYPT 2013 [JN13] and PKC 2014 [Kra14]. Full details on these works are available in Part II of this manuscript.

An article cosigned with Michael Naehrig was published at AFRICACRYPT 2014 [PV14], and its proof-of-concept implementation is openly available. This article is not detailed in this manuscript, we therefore provide its abstract for completeness.

[LP13] **On the Minimal Number of Bootstrappings in Homomorphic Circuits.**
T. Lepoint, P. Paillier. (WAHC 2013)

[CCK+13] **Batch Fully Homomorphic Encryption over the Integers.**
J.H. Cheon, J.-S. Coron, J. Kim, M.S. Lee, T. Lepoint, M. Tibouchi, A. Yun. (EUROCRYPT 2013)
Full version avalable at [CLT13a].

[CLT14a] **Scale-Invariant Fully Homomorphic Encryption over the Integers.**
J.-S. Coron, T. Lepoint, M. Tibouchi. (PKC 2014)
Full version avalable at [CLT14b].

[LN14a] **A Comparison of the Homomorphic Encryption Schemes FV and YASHE.**
T. Lepoint, M. Naehrig. (AFRICACRYPT 2014)
Full version available at [LN14b].

We conduct a theoretical and practical comparison of two Ring-LWE-based, scale-invariant, leveled homomorphic encryption schemes – Fan and Vercauteren's adaptation of BGV and the YASHE scheme proposed by Bos, Lauter, Loftus and Naehrig. In particular, we explain how to choose parameters to ensure correctness and security against lattice attacks. Our parameter selection improves the approach of van de Pol and Smart to choose parameters for schemes based on the Ring-LWE problem by using the BKZ-2.0 simulation algorithm.

We implemented both encryption schemes in C++, using the arithmetic library FLINT, and compared them in practice to assess their respective strengths and weaknesses. In particular, we performed a homomorphic evaluation of the lightweight block cipher SIMON. Combining block ciphers with homomorphic encryption allows to solve the gargantuan ciphertext expansion in cloud applications.

[Lep14] **A proof-of-concept implementation of the homomorphic evaluation of SIMON using FV and YASHE leveled homomorphic cryptosystems.**
T. Lepoint.

Multilinear Maps. In collaboration with Jean-Sébastien Coron (from University of Luxembourg) and Mehdi Tibouchi (from NTT Secure Platform Laboratories), we designed the second multilinear maps scheme candidate based on the breakthrough result of Garg, Gentry and Halevi [GGH13a]. We also provide the first implementation of such a scheme, which appears to be arguably practical, as a 7-partite (resp. 26-partite) one-round key exchange protocol runs in a matter of seconds (resp. minutes).

An article was published at CRYPTO 2013 [CG13a], and the associated proof-of-concept implementation is openly available. Full details are provided in Part III of this manuscript.

[CLT13b] **Practical Multilinear Maps over the Integers.**
J.-S. Coron, T. Lepoint, M. Tibouchi. (CRYPTO 2013)
Full version available at [CLT13c].

[Lep13] **An Implementation of Multilinear Maps over the Integers.**
T. Lepoint.

White-Box Cryptography. In collaboration with Cécile Delerablée, Pascal Paillier and Matthieu Rivain (from CryptoExperts), we worked on white-box cryptography. In particular, with Matthieu Rivain we designed a very efficient attack (of complexity 2^{22}) against the last supposedly secure white-box AES implementation. A second work translated the folklore intuitions behind white-box cryptography (used to design all the broken white-box candidates) into concrete security notions.

Overall, our results shed more light on the different aspects of white-box security and provide concrete constructions that achieve them in a provable fashion.

Two articles were published at SAC 2013 [LLL13] (the first article was merged with an independent work of Y. De Mulder, P. Roelse and B. Preneel). These articles are briefly discussed in Appendix A, including some comments on the relation between white-box cryptography and indistinguishability obfuscation [GGH+13b].

[LRM+13] **Two Attacks on a White-Box AES Implementation.**
T. Lepoint, M. Rivain, Y. De Mulder, P. Roelse, B. Preneel. (SAC 2013)
Full version available at [LR13].

[DLPR13b] **White-Box Security Notions for Symmetric Encryption Schemes.**
C. Delerablée, T. Lepoint, P. Paillier, M. Rivain. (SAC 2013)
Full version available at [DLPR13a].

Chapter 3

Preliminaries

In this chapter we recall some preliminary notions we are going to use throughout the entire thesis. We first start by giving some guideline for the notation used in the manuscript. Next we briefly introduce some background about lattices. Finally, we recall two lemmas that will be – repeatedly – used in the different parts of the thesis: the Leftover Hash Lemma and the Rejection Sampling Lemma.

3.1 Notation

Throughout the manuscript, we tried to make our notation uniform. We denote the set of real numbers by $\mathbb{R}$, the set of integers by $\mathbb{Z}$ and the set of non-negative integers by $\mathbb{N}$. We denote by $\mathbb{Z}_n$ the ring $\mathbb{Z}/n\mathbb{Z}$ of integers modulo an integer n; when n is prime, we denote by $\mathbb{F}_n = \mathbb{Z}_n$ the field with n elements. For coprime integers p_i's and integers a_i's, we denote by $\mathsf{CRT}_{p_1,\ldots,p_\ell}(a_1,\ldots,a_\ell)$ the unique integer u smaller than $\prod_{i=1}^{\ell} p_i$ such that $u \bmod p_i = a_i$ for all $1 \leqslant i \leqslant \ell$.

We denote by $a \leftarrow S$ the action of picking a independently and uniformly at random from some set S, and by $a \leftarrow \mathcal{R}(\cdots)$ the action of running algorithm $\mathcal{R}$ on some inputs and naming a the value returned by $\mathcal{R}$. If a set S is finite, we denote by $\mathcal{U}(S)$ the uniform distribution on S. Also the probability that an event X occurs is denoted by $\Pr[X]$.

We use the standard Landau notation $o(\cdot)$, $\mathcal{O}(\cdot)$, $\Omega(\cdot)$. We also use the notation $\tilde{\mathcal{O}}(\cdot)$ for hiding poly-logarithmic factors, *i.e.* $f(n) = \tilde{\mathcal{O}}(g(n)) = \mathcal{O}(g(n)\log^c(n))$ for some fixed constant c. We let $\mathsf{poly}(n)$ denote an unspecified function $f(n) = \mathcal{O}(n^c)$ for some constant c. A negligible function, denoted generically by $\mathsf{negl}(n)$, is a function f that decreases faster than n^{-c} for any constant $c > 0$. We say that a function is overwhelming if it is $1 - \mathsf{negl}(n)$.

Vectors and Matrices. We denote vectors (resp. matrices) by bold lower (resp. upper) case roman letters, such as $\mathbf{x}$ (resp. $\mathbf{A}$). The ith coefficient of $\mathbf{x}$ will be denoted x_i and the (i,j)-th coefficient of $\mathbf{A}$ will be denoted a_{ij}. By convention, vectors are assumed to be in column form and for a vector $\mathbf{x}$ (resp. a matrix $\mathbf{A}$), we denote $\mathbf{x}^t$ (resp. $\mathbf{A}^t$) the transpose of $\mathbf{x}$ (resp. of $\mathbf{A}$). When a square matrix $\mathbf{A}$ is invertible, we denote by $\mathbf{A}^{-1}$ its inverse. Particular values we will use are $\mathbf{0} = (0,\ldots,0)^t$, $\mathbf{1} = (1,\ldots,1)^t$ and $\mathbf{I}$ the identity matrix

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}.$$

We denote $\mathrm{GL}_n(\mathbb{R})$ (resp. $\mathrm{GL}_n(\mathbb{Z})$) the set of invertible matrices over $\mathbb{R}$ (resp. over $\mathbb{Z}$). Note that if $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$, we have $\det(\mathbf{U}) = \pm 1$.

If two vectors $\mathbf{x}, \mathbf{y}$ have matching dimensions, we denote their inner product by $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i$. For a vector $\mathbf{x} \in \mathbb{R}^n$ and $p \in [1, +\infty)$, we define the ℓ_p norm as

$$\|\mathbf{x}\|_p = \Big(\sum_{i=1}^{n} |x_i|^p \Big)^{1/p},$$

and for $p = \infty$, we define the ℓ_∞ norm $\|\mathbf{x}\|_\infty = \max_{i=1}^n |x_i|$. When p is not specified, $\|\mathbf{x}\|$ is assumed to represent the ℓ_2 norm of $\mathbf{x}$ (*i.e.* its Euclidean norm). The operator norm ℓ_p of a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, for $p \in [1, +\infty]$, is defined by $\|\mathbf{A}\|_p = \sup_{\mathbf{x} \neq \mathbf{0}} \|\mathbf{A}\mathbf{x}\|_p / \|\mathbf{x}\|_p$.

Random Experiments. A random experiment is an interactive protocol played by a group of probabilistic algorithms interacting together. Random experiments are also referred to as (probabilistic) games and are expressed as just a list of actions involving the players. We denote by

$$\Pr\left[\mathsf{action}_1 \text{ then } \mathsf{action}_2 \text{ then } \ldots \text{ then } \mathsf{action}_n : \mathsf{event}\right]$$

the probability that event occurs after executing $\mathsf{action}_1, \ldots, \mathsf{action}_n$ in sequential order, the probability being taken over the probability spaces of all the random variables involved in these actions. One often refers to those as the random coins of the game $(\mathsf{action}_1, \ldots, \mathsf{action}_n)$.

Concrete Reductions. An algorithm $\mathcal{R}$ is said to $(\tau_\mathcal{A}, \varepsilon_\mathcal{A}, \tau_\mathcal{R}, \varepsilon_\mathcal{R})$-reduce a problem P_1 to a problem P_2, which we then denote by $P_1 \Leftarrow_\mathcal{R} P_2$, if $\mathcal{R}$ solves P_1 with probability at least $\varepsilon_\mathcal{R}$ using an algorithm $\mathcal{A}$ solving P_2 with probability $\varepsilon_\mathcal{A}$ as an oracle, and in time at most $\tau_\mathcal{R}$. By time at most $\tau_\mathcal{R}$, we mean that $\mathcal{R}$ runs in at most $\tau_\mathcal{R}$ elementary time units, oracle calls to $\mathcal{A}$ counting for exactly $\tau_\mathcal{A}$ time units. $\mathcal{R}$ is then called a *reduction* from P_1 to P_2, or more precisely a $(\tau_\mathcal{A}, \varepsilon_\mathcal{A}, \tau_\mathcal{R}, \varepsilon_\mathcal{R})$-reduction.

In the asymptotic setting, one says that a reduction $\mathcal{R}$ is polynomial when $\tau_R / \varepsilon_\mathcal{R}$ is polynomial in some complexity parameter if $\tau_\mathcal{A} / \varepsilon_\mathcal{A}$ was polynomial in the first place. When there exists $\mathcal{R}$ such that $P_1 \Leftarrow_\mathcal{R} P_2$ and $\mathcal{R}$ is polynomial, we may simply write $P_1 \Leftarrow P_2$.

3.2 A refresher on Lattices

In this section, we briefly recall some necessary mathematical background on lattices.

3.2.1 Lattices

A *lattice* L is a discrete additive subgroup of $\mathbb{R}^n$. Equivalently, a lattice can be defined as the set of all linear integer combinations of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$, and we write

$$L = \mathbf{b}_1\mathbb{Z} \oplus \cdots \oplus \mathbf{b}_d\mathbb{Z} = \Big\{ \sum_{i=1}^{d} x_i \mathbf{b}_i : \mathbf{x} = (x_1, \ldots, x_d)^t \in \mathbb{Z}^d \Big\}.$$

For the sake of simplicity, we restrict ourselves to full rank lattices, *i.e.* lattices such that $d = n$. We say that the set $\{\mathbf{b}_i\}_{i=1}^n$ forms a *basis* of the lattice that its elements span; and we define the basis matrix $\mathbf{B}$ as the matrix whose columns are the $\mathbf{b}_i$'s. For $n \geqslant 2$, a lattice has infinitely many bases, of the same cardinality n. The determinant (or volume) of a lattice is defined as

$$\det(L) = \big(\det(\mathbf{B}\mathbf{B}^t) \big)^{1/2} = |\det(\mathbf{B})|,$$

where $\mathbf{B}$ is any basis of L. This quantity is well-defined since it is independent of the choice of the basis: if $\mathbf{B}'$ is another basis of L, then there exists a unimodular matrix $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$ such that $\mathbf{B}' = \mathbf{B} \cdot \mathbf{U}$. Figure 3.1 gives a two-dimensional lattice with two different bases. Throughout the thesis, we will indifferently refer to a lattice, to one of its bases or to one of its basis matrices as the lattice itself.

Any lattice L has several popular lattice invariants, *i.e.* intrinsic values independent of the particular representation of L. This includes, among others, the determinant of the lattice and the successive minima of the lattice defined by $\lambda_i(L) = \min(r : \dim \mathrm{Span}(L \cap \mathcal{B}_n(\mathbf{0}, r)) \geqslant i)$ for $i \leqslant n$, where $\mathcal{B}_n(\mathbf{0}, r)$ refers to the n-dimensional closed ball of center $\mathbf{0}$ and radius r. The minimum of L is $\lambda_1(L)$, that is the (Euclidean) norm of a shortest non-zero vector of L.

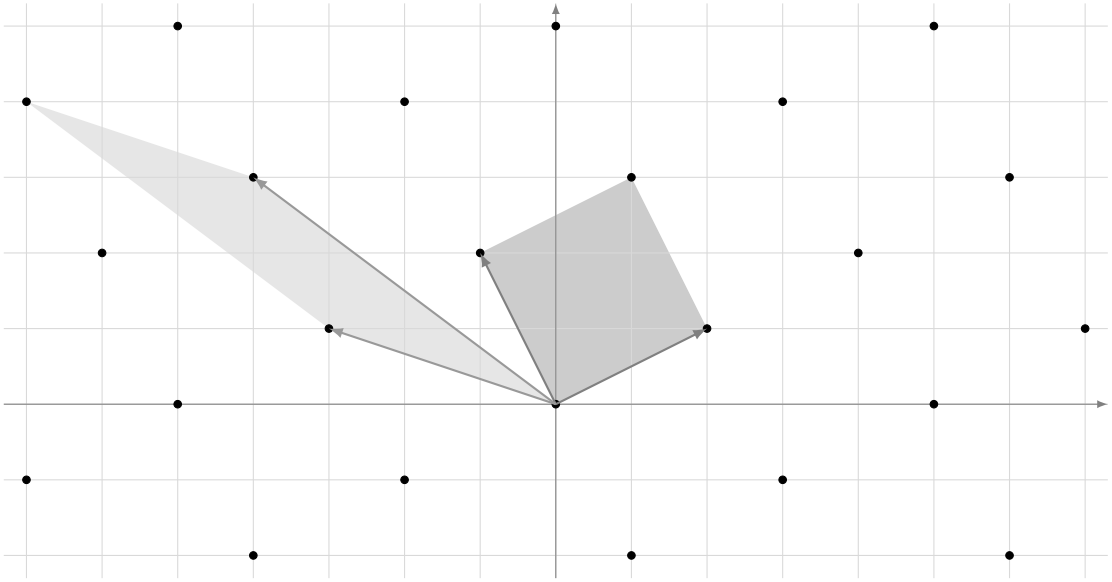


Figure 3.1 – A two dimensional lattice along with two of its bases, and its volume.

Lattice Reduction. Among all the bases of a lattice L, some are 'better' than others. The goal of lattice basis reduction is to find another basis of the same lattice with guaranteed norm and orthogonality properties. Any basis $\mathbf{B} = (\mathbf{b}_1|\dots|\mathbf{b}_n)$ can be uniquely written as $\mathbf{B} = \mathbf{B}^* \cdot \mathbf{R}$ where $\mathbf{B}^*$ is an orthogonal matrix and $\mathbf{R}$ is upper triangular with diagonal coefficients equal to 1. We call $\mathbf{B}^* = (\mathbf{b}_1^*|\cdots|\mathbf{b}_n^*)$ the Gram-Schmidt orthogonalization of $\mathbf{B}$. Note that the Gram-Schmidt orthogonalization provides useful information on the lattice invariants of a lattice L. In particular, if $\mathbf{B}$ is a basis matrix of L, we have

$$\det(L) = \prod_{i=1}^{n} \|\mathbf{b}_i^*\| \quad \text{and} \quad \min_{j \geqslant i} \|\mathbf{b}_j^*\| \leqslant \lambda_i(L) \leqslant \max_{j \leqslant i} \|\mathbf{b}_j^*\|, \text{ for all } i \leqslant n\,.$$

Following the approach popularized by Gama and Nguyen [GN08], we say that a specific basis $\mathbf{B}$ has *root Hermite factor* δ if its element of smallest norm $\mathbf{b}_1$ (*i.e.* we assume that basis vectors are ordered by their norm) satisfies

$$\|\mathbf{b}_1\| = \delta^n \cdot |\det(\mathbf{B})|^{1/n}\,.$$

A classical lattice basis reduction algorithm is LLL (due to Lenstra, Lenstra and Lovász [LLL82]). The LLL algorithm runs in polynomial time and provides bases of quite decent quality. For many cryptanalytic applications, Schnorr and Euchner's blockwise algorithm BKZ [SE94] is the most practical algorithm for lattice basis reduction in high dimensions. It provides bases of higher quality but its running time increases significantly with the blocksize. Now if A denotes a lattice basis reduction algorithm, applying it to $\mathbf{B}$ yields a reduced basis $\mathbf{B}' \leftarrow \mathsf{A}(\mathbf{B})$. Thus we can define $\delta_{\mathsf{A}(\mathbf{B})}$ as the value such that

$$\|\mathbf{b}_1'\| = \delta_{\mathsf{A}(\mathbf{B})}^n \cdot |\det(\mathbf{B}')|^{1/n} = \delta_{\mathsf{A}(\mathbf{B})}^n \cdot |\det(\mathbf{B})|^{1/n}\,.$$

It is conjectured [GN08, CN11] that the value $\delta_{\mathsf{A}(\mathbf{B})}$ depends mostly on the lattice basis reduction algorithm, and not on the input basis $\mathbf{B}$ (unless it has a special structure, unusually short vectors or cannot be considered random). Thus, in this thesis, we refer to this value as δ_{A}. For example for LLL and BKZ-20 (*i.e.* BKZ with a blocksize $\beta = 20$), in the literature one can find the well-known values $\delta_{\mathsf{LLL}} \approx 1.021$ and $\delta_{\mathsf{BKZ\text{-}20}} \approx 1.013$.

Schnorr and Euchner's blockwise algorithm BKZ [SE94] takes as input parameter the blocksize β, which impacts both the running time and the quality of the resulting basis. In [GN08], it is mentioned that BKZ-β for $\beta > 30$ for non trivial dimensions does not terminate in reasonable time. In 2011, Chen and Nguyen described an implementation of BKZ called BKZ-2.0 [CN11] (see

also the full version of the paper in [CN13]). This implementation incorporates some of the latest improvements of lattice basis reduction: preprocessing of local bases (Kannan [Kan83]), shorter enumeration radius, early abort (Hanrot, Pujot and Stehlé [HPS11]), extreme pruning (Gama, Nguyen and Regev [GNR10]). In particular implementing these techniques allows to consider blocksizes $\beta \geqslant 50$. We refer the interested reader to [CN11, CN13] for additional information.

The algorithm BKZ-2.0$_{N,\beta}$ is parametrized by two parameters: the *maximal* number of rounds N and the blocksize β, and takes as input an LLL-reduced m-dimensional basis. The rough idea is that in each round, it iterates over an index $i \leqslant m - \beta$, considers the β-dimensional lattice spanned by the current basis vectors $\mathbf{b}_i, \ldots, \mathbf{b}_{i+\beta-1}$ and projects it onto the orthogonal complement of the first $i-1$ basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. It then performs an enumeration using extreme pruning on this projected lattice to find the shortest vector, and this vector is inserted into the main lattice basis at the ith position. Note that BKZ-2.0$_{N,\beta}$ can be aborted before reaching the Nth round if the basis has not been modified in the current round, *i.e.* a fix point has been attained.

Building upon the analysis of [HPS11], Chen and Nguyen provided an *efficient simulation algorithm* to model the behavior of BKZ-2.0 in high dimensions with large blocksizes $\geqslant 50$.[1] The simulation algorithm takes as input the Gram-Schmidt norms of an LLL-reduced basis, a blocksize $\beta \in \{50, \ldots, m\}$ and a number N of rounds, and outputs a prediction for the Gram-Schmidt norms after N rounds of BKZ-β reduction. A Python implementation of this simulation algorithm has been made available by the authors in [CN13].

3.2.2 Average-Case Problems and Algorithmic Problems on Lattices

Almost all modern lattice-based cryptography is founded on two average-case computational problems: the Short Integer Solution (SIS), and the Learning With Errors (LWE). A strong argument in favor of lattice-based cryptography is that these average-case problems are connected to worst-case lattice algorithmic problems extensively studied in the literature.

Algorithmic Problems on Lattices. The most studied algorithmic problems on lattices are computational problems related to the abovementioned lattice invariants. We briefly mention some of these problems below (for integer lattices); but precise descriptions of these problems are not the purpose of this thesis.

SVP$_\gamma$. The Shortest Vector Problem with approximation factor γ is as follows: Given a n-dimensional lattice L and one of its bases $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, find a vector $\mathbf{c} \in L$ such that $0 < \|\mathbf{c}\| \leqslant \gamma \cdot \lambda_1(L)$.

GapSVP$_\gamma$. The Gap Shortest Vector Problem with approximation factor γ is as follows: Given a n-dimensional lattice L, one of its bases $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and a number d, decide if $\lambda_1(L) \leqslant d$ or $\lambda_1(L) \geqslant \gamma \cdot d$.

SIVP$_\gamma$. The Shortest Independent Vectors Problem with approximation factor γ is as follows: Given a n-dimensional lattice L and one of its bases $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, find n linearly independent vectors $\mathbf{c}_i \in L$ for $i \leqslant n$ such that $\max_i \|\mathbf{c}_i\| \leqslant \gamma \cdot \lambda_n(L)$.

CVP$_\gamma$. The Closest Vector Problem with approximation factor γ is as follows: Given a n-dimensional lattice L, one of its bases $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and a target vector $\mathbf{t} \in \mathbb{R}^n$, find a vector $\mathbf{c} \in L$ such that $0 < \|\mathbf{c} - \mathbf{t}\| \leqslant \gamma \cdot \mathrm{dist}(\mathbf{t}, L) = \gamma \cdot \inf_{\mathbf{e} \in L} \|\mathbf{e} - \mathbf{t}\|$.

BDD$_\gamma$. The Bounded Distance Decoding Problem with approximation factor γ is as follows: Given a n-dimensional lattice L, one of its bases $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and a target vector $\mathbf{t} \in \mathbb{R}^n$ such that $\mathrm{dist}(\mathbf{t}, L) \leqslant \gamma^{-1} \cdot \lambda_1(L)$, find a vector $\mathbf{c} \in L$ such that $\|\mathbf{c} - \mathbf{t}\| = \mathrm{dist}(\mathbf{t}, L)$.

It is worth noting that it is not currently known how to significantly[2] exploit quantum computing to solve these problems more efficiently than with classical algorithms. This is why lattice-based

[1] This simulation algorithm is an *ideal* simulation procedure [CN13]. In particular, it assumes that the probability of success of extreme pruning is $p \approx 1$ and it does not model the behavior for blocksizes $\beta < 50$ correctly.

[2] Using Grover's quantum search algorithm [Gro96] might end up reducing the constants in the exponent, but the overall complexity remains essentially the same [Lud03, LMP13].

cryptography, which eventually relies on these problems, is said to withstand quantum attacks – it is one of the favorite candidates for post-quantum cryptography. However, nothing excludes that (to-be-discovered) quantum algorithms could dramatically change this situation.

From the definitions of the problems, it is clear that their complexity increases as the dimension n increases, and decreases as the approximation factor γ increases. Let us recall some computational results mentioned in the survey [Reg10b] – we refer the interested reader to this survey for all interesting references. By the LLL algorithm (and subsequent improvements), we are able to efficiently (*i.e.* in polynomial time) approximate lattice problems to within exponential factors, namely $\gamma(n) = 2^{n \log\log n / \log n}$. On the other hand, we know that there exists $c > 0$ for which approximating lattice problem to within $\gamma(n) = n^{c/\log\log n}$ is hard[3], unless some unlikely events occur (such as $\mathsf{NP} \not\subset \mathsf{RSUBEXP}$ [Reg10b]). Between these two extreme approximation factors, there is a wide range of complexity possibilities. In particular, constructions in lattice-based cryptography rely on the hardness of these algorithmic algorithms for an approximation factor $\gamma(n) = \mathsf{poly}(n)$. The best known algorithms in this latter case all have exponential complexity bounds and are believed to be at least exponential-time in the worst case.

In general, cryptography is based on average case problems, and lattice-based cryptography does not escape this paradigm. Let us present the SIS and LWE problems below.

SIS. The SIS problem was introduced in the seminal work of Ajtai [Ajt96] showing connections between worst-case lattice problems and the average-case SIS problem. Let n and $q = \mathsf{poly}(n)$ be integers, and let $\beta > 0$. The $\mathrm{SIS}_{q,n,m,\beta}$ problem consists in, given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for some $m = \mathsf{poly}(n)$, finding a non-zero integer vector $\mathbf{z}$ such that $\mathbf{A}\mathbf{z} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\| \leqslant \beta$. By the pigeonhole principle, if $\beta \geqslant \sqrt{m} q^{n/m}$ then the SIS instances are guaranteed to have a solution. Using Gaussian techniques, Micciancio and Regev [MR07] improved Ajtai's result to show that, for a large enough q as a function of n and β, the $\mathrm{SIS}_{q,n,m,\beta}$ problem is as hard (on the average) as the $\tilde{\mathcal{O}}(\sqrt{n}\beta)$-SIVP problem for *all* lattices of dimension n.

In 2006, a ring variant of SIS was introduced independently by Peikert and Rosen [PR06] and Lyubashevsky and Micciancio [LM06]. If we restrict to the rings $R = \mathbb{Z}[x]/(x^n + 1)$ for n a power of 2, and $q = \mathsf{poly}(n)$ is an integer, the (search) $\mathrm{RSIS}_{q,n,m,\beta}$ problem is: Given $\mathbf{a}_1, \ldots, \mathbf{a}_m \leftarrow \mathbb{Z}_q[x]/(x^n + 1)$, find random $\mathbf{z}_1, \ldots, \mathbf{z}_m \in \mathbb{Z}[x]/(x^n + 1)$ such that $\sum_{i=1}^m \mathbf{a}_i \cdot \mathbf{z}_i = \mathbf{0} \bmod q$ and $0 < \|\mathbf{Z}\| \leqslant \beta$ where $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_m)^t$.

In [LM06] it was shown that if $R = \mathbb{Z}[x]/(x^n + 1)$, where n is a power of 2, then the $\mathrm{RSIS}_{q,n,m,\beta}$ problem is as hard as the $\tilde{\mathcal{O}}(\sqrt{n}\beta)$-SVP problem in all lattices that are ideals in R.

LWE. The LWE problem was introduced in the seminal work of Regev [Reg09]. For the same parameters n and q, and $\alpha \in (0, 1)$, the search $\mathrm{LWE}_{q,n,m,\alpha}$ problem is: Given a "noisy" random linear system $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{b} = \mathbf{A}^t \cdot \mathbf{s} + \mathbf{e}$ where $\mathbf{A}$ is uniformly random and the entries of $\mathbf{e}$ are independent and identically distributed according to a centered discrete Gaussian distribution over the integers[4] of standard deviation $\alpha \cdot q$, to recover the secret vector $\mathbf{s} \in \mathbb{Z}_q^n$.

The following theorem subsumes the latest security reductions [Reg09, BLP+13]:

Theorem 3.1 (As stated in [LLS14])**.** *Let $m, q(n) \geqslant 2$ and $\alpha(n) \in (0, 1)$ be such that $\alpha \cdot q \geqslant 2\sqrt{n}$. If $q = \mathsf{poly}(n)$ is prime, there exists a* quantum *polynomial reduction from SIVP_γ in dimension n to $\mathrm{LWE}_{q,n,m,\alpha}$ with $\gamma = \tilde{\mathcal{O}}(n/\alpha)$. For any q, there exists a* classical *polynomial time reduction from GapSVP_γ in dimension $\Theta(\sqrt{n})$ to $\mathrm{LWE}_{q,n,m,\alpha}$ with $\gamma = \tilde{\mathcal{O}}(n^2/\alpha)$.*

In 2010, a ring variant of LWE was introduced by Lyubashevsky, Peikert and Regev [LPR13a]. The most general definition requires some algebraic tools that are beyond the scope of this thesis. If we restrict to the rings $R = \mathbb{Z}[x]/(x^n + 1)$ for n a power of 2, $q = \mathsf{poly}(n)$ is an integer and $\alpha \in (0, 1)$, the (search) $\mathrm{RLWE}_{q,n,m,\alpha}$ problem is: Given $m = \mathsf{poly}(n)$ samples of the form $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + \mathbf{e})$ where

[3] Note that known results differ on the problem and the underlying norm. For example, SVP is NP-hard to approximate in the ℓ_∞ norm to within $n^{c/\log\log n}$ for a $c > 0$, while this holds for CVP in the ℓ_p norm for all $1 \leqslant p \leqslant \infty$.

[4] We will define more precisely the discrete Gaussian distribution in Chapter 4. Here we consider the probability distribution $\exp(-|z|^2/(2\sigma^2))$ for $z \in \mathbb{Z}$, where σ is the standard deviation.

$\mathbf{a} \leftarrow \mathbb{Z}_q[x]/(x^n+1)$ and the coefficients of $\mathbf{e} \in \mathbb{Z}_q[x]/(x^n+1)$ are independent and identically distributed according to a centered discrete Gaussian distribution over the integers of standard deviation $\alpha \cdot q$, recover the secret polynomial $\mathbf{s} \in \mathbb{Z}_q[x]/(x^n+1)$.

Throughout the thesis, we will not use LWE – this explains its brief presentation. However, it is worth noting that its versatility makes this problem a *cornerstone* of efficient and powerful primitives such as Fully Homomorphic Encryption [BGV12], Attribute-Based Encryption for all circuits [GVW13], and *numerous* others applications. In Part I of this thesis, we will slightly generalize the SIS problem to construct our efficient lattice-based signature scheme. For details on SIS and LWE, we refer the readers to [MR09, Reg10a, LS12, LPR13a, BLP+13, LPR13b, MP13, LLS14] among other works.

3.3 Useful Lemmas

In this section, we recall the Leftover Hash Lemma (and provide some corollaries) and the Rejection Sampling method, both being extensively used throughout the whole thesis.

3.3.1 Leftover Hash Lemma

In this section, we recall the classical Leftover Hash Lemma [HILL99] and give two easy corollaries at the heart of all our constructions.

First, we say that a distribution D is ε-uniform if its statistical distance from the uniform distribution is at most ε, where the statistical distance $\Delta(D_1, D_2)$ between two distributions D_1, D_2 over a finite domain X is given by $\Delta(D_1, D_2) = \frac{1}{2}\sum_{x \in X} |D_1(x) - D_2(x)|$.

Let X and Y be finite sets. A family $\mathcal{H}$ of hash functions from X to Y is said to be *pairwise-independent* if for all distinct $x, x' \in X$, $\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(x')] = 1/|Y|$.

Lemma 3.2 (Leftover Hash Lemma [HILL99])**.** *Let $\mathcal{H}$ be a family of pairwise hash functions from X to Y. Suppose that $h \leftarrow \mathcal{H}$ and $x \leftarrow X$ are chosen uniformly and independently. Then, $(h, h(x))$ is $1/2\sqrt{|Y|/|X|}$-uniform over $\mathcal{H} \times Y$.*

One can then deduce the following corollary for random subset sums over a finite abelian group.

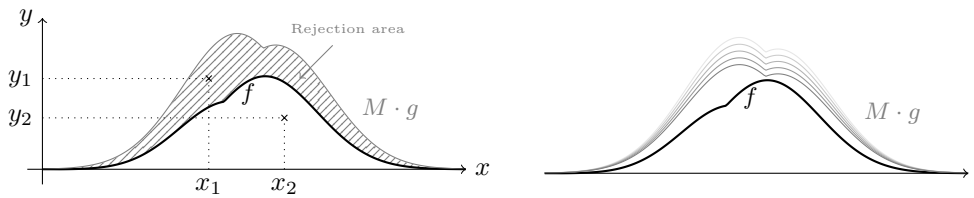
Corollary 3.3. *Let $m \geqslant 2$. Let G be a finite abelian group. Set $x_1, \ldots, x_m \leftarrow G$ uniformly and independently, set $s_1, \ldots, s_m \leftarrow \{0,1\}$, and set $y = \sum_{i=1}^m s_i x_i \in G$. Then $(x_1, \ldots, x_m, y)$ is $1/2\sqrt{|G|/2^m}$-uniform over G^{m+1}.*

Proof. We consider the following hash function family $\mathcal{H}$ from $\{0,1\}^m$ to G. Each member $h \in \mathcal{H}$ is parameterized by the elements $(x_1, \ldots, x_m) \in G^m$. Given $\mathbf{s} \in \{0,1\}^m$, we define $h(\mathbf{s}) = \sum_{i=1}^m s_i \cdot x_i \in G$. The hash function family is clearly pairwise independent. Therefore by Lemma 3.2, $(h, h(\mathbf{s}))$ is $1/2\sqrt{|G|/2^m}$-uniform over G^{m+1}. □

Similarly, one can also deduce the following corollary for finite linear combinations modulo a prime $q > 2^\alpha$, where the s_i are α-bit integers instead of bits:

Corollary 3.4. *Let $m \geqslant 2$. Set $x_1, \ldots, x_m \leftarrow \mathbb{Z}_q$ uniformly and independently, set $s_1, \ldots, s_m \leftarrow (-2^\alpha, 2^\alpha)$, and set $y = \sum_{i=1}^m s_i \cdot x_i \bmod q$. Then $(x_1, \ldots, x_m, y)$ is $1/2\sqrt{q/2^{(\alpha+1)\cdot m}}$-uniform over $\mathbb{Z}_q^{m+1}$.*

Proof. Let us consider the hash function family $\mathcal{H}$ from $(-2^\alpha, 2^\alpha)^m$ to $\mathbb{Z}_q$. Each member $h \in \mathcal{H}$ is parameterized by the element $(x_1, \ldots, x_m) \in \mathbb{Z}_q^m$. Given $\mathbf{s} \in (-2^\alpha, 2^\alpha)^m$, we define $h(\mathbf{s}) = \sum_{i=1}^m s_i \cdot x_i \in \mathbb{Z}_q$. The hash function family is clearly pairwise independent since q is prime. Therefore by Lemma 3.2, $(h, h(\mathbf{s}))$ is $1/2\sqrt{q/2^{(\alpha+1)\cdot m}}$-uniform over $\mathbb{Z}_q^{m+1}$. □



(a) (x_i, y_i) is sampled uniformly in the area under $M \cdot g$, and accepted when $y_i \leqslant f(x_i)$

(b) M can be reduced when g is better adapted to f

Figure 3.2 – Rejection sampling from the distribution of g to get the distribution of f

3.3.2 Rejection Sampling

In this section, we give an overview of the rejection sampling technique (that will be at the heart of all the chapters of Part I).

Rejection sampling is a well-known method introduced by von Neumann [vN51] to sample from an arbitrary target probability distribution f, given a source bound to a different probability distribution g. Conceptually, the method works as follows. A sample x is drawn from g and is accepted with probability $f(x)/(M \cdot g(x))$, where M is some positive real. If it is not accepted, then the process is restarted. It is not hard to prove that if $f(x) \leqslant M \cdot g(x)$ for all x, then the rejection sampling procedure produces exactly the distribution of f. Furthermore, because the expected number of times the procedure will need to be restarted is M, it is crucial to keep M as small as possible, possibly by tailoring the function g so that it resembles the target function f as much as possible. In particular, since rejection sampling can be interpreted as sampling a random point (x_i, y_i) in the area under the distribution $M \cdot g$ (see Figure 3.2) and accepting if and only if $y_i \leqslant f(x_i)$, reducing the area between the two curves will reduce M.

Lemma 3.5 (Rejection Sampling)**.** *Let V be an arbitrary set, and $h\colon V \to \mathbb{R}$ and $f\colon \mathbb{Z}^m \to R$ be probability distributions. If $g_v\colon \mathbb{Z}^m \to R$ is a family of probability distributions indexed by $v \in V$ with the property that there exists a $M \in \mathbb{R}$ such that*

$$\forall v \in V, \forall \mathbf{z} \in \mathbb{Z}^m, M \cdot g_v(\mathbf{z}) \geqslant f(\mathbf{z}) \ ,$$

then, the output distributions of the following two algorithms are identical:

1. *$v \leftarrow h$, $z \leftarrow g_v$, output $(\mathbf{z}, v)$ with probability $f(\mathbf{z})/\big(M \cdot g_v(\mathbf{z})\big)$.*
2. *$v \leftarrow h$, $z \leftarrow f$, output $(\mathbf{z}, v)$ with probability $1/M$.*

Part One

Design and Implementation of a Lattice-Based Signature Scheme

Overview

Lattice-based cryptography is arguably the most promising replacement for standard cryptography, in the event quantum computers become a threat. Its sound hardness results and the versatility of its average-case problems (*e.g.* SIS and LWE) make it an active research area [MR09]. Researchers are rapidly discovering new lattice-based primitives, such as fully-homomorphic encryption [Gen09], multi-linear maps [GGH13a], attribute-based encryption [GVW13], and indistinguishability obfuscation [GGH$^+$13b] that had no previous constructions based on classical number-theoretic techniques.

It is often said that lattice-based cryptosystems are efficient and easy to implement, as basic operations are only matrix-vector multiplications modulo an integer q. Lattice-based public-key cryptographic primitives such as encryption schemes [HPS98, LPR13a, PG13] and digital signatures [Lyu12, GLP12, GOPS13, BB13] already have somewhat practical lattice-based instantiations. However, the reality of their implementation is slightly more intricate than this simplified point of view. While lattice-based cryptography is asymptotically efficient (notably when using ideal lattices), in practice only a few implementation results for lattice-based primitives are described, and the resulting key/ciphertext/signature sizes and performances are often rather unsatisfactory.

This part of the thesis presents contributions to two areas of lattice-based cryptography. First, in Chapter 4 we propose an efficient discrete Gaussian sampling algorithm over the integers, building block of numerous lattice-based cryptosystems, that is compatible with constrained devices. Second, we improve over today's supposedly most efficient lattice-based signature schemes and propose concrete instantiations. The resulting proof-of-concept implementation illustrates the practicality of lattice-based cryptography and compares very favorably to the classical algorithms RSA and ECDSA. Chapters 5 and 6 are devoted to our works on this topic, some of which yielded interesting follow-up works [PG14, OPG14].

Chapter 4

Efficient Discrete Gaussian Sampling over the Integers

4.1 Introduction

This chapter introduces an efficient algorithm to sample according to a discrete Gaussian distribution over the integers, with small storage footprint and no transcendental function evaluation. All the operations of our algorithm are very simple and require only small integer arithmetic. As a consequence, we show that discrete Gaussian sampling over the integers, a fundamental building block of lattice-based cryptography, *is* appropriate for use in constrained devices.

This chapter was part of the article Lattice Signature and Bimodal Gaussian [DDLL13a], cosigned with L. Ducas, A. Durmus and V. Lyubashevsky and published at Crypto 2013 [CG13a]. The full version of the article is available at [DDLL13b].

Background. To prevent the leakage of the signer's secret key in the early attempts of lattice-based signature schemes [GGH97, HHGP+03], Gentry, Peikert and Vaikuntanathan proposed in 2008 to use discrete Gaussian sampling [GPV08]. Indeed, sampling solutions in [GGH97] according to a discrete Gaussian distribution leaks no information about the geometry of the lattice.

Let $L \subset \mathbb{R}^n$ be a full-rank lattice. The discrete Gaussian distribution $D_{L,\mathbf{c},\sigma}$ of support L, center $\mathbf{c} \in \mathbb{R}^n$ and standard deviation σ is defined by:

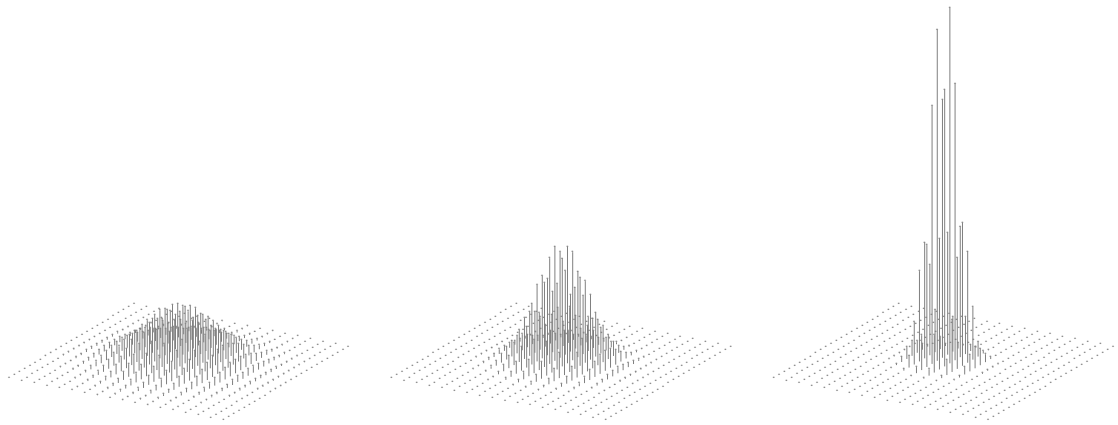
$$\forall \mathbf{x} \in L, D_{L,\mathbf{c},\sigma}(\mathbf{x}) = \frac{\rho_{\mathbf{c},\sigma}(\mathbf{x})}{\sum_{\mathbf{y}\in L} \rho_{\mathbf{c},\sigma}(\mathbf{y})} ,$$

where $\rho_{\mathbf{c},\sigma}(\mathbf{x}) = \exp(-\|\mathbf{x}-\mathbf{c}\|^2/(2\sigma^2))$.[1] The subscripts L and $\mathbf{c}$ will be omitted when $L = \mathbb{Z}^n$ and $\mathbf{c} = \mathbf{0}$ respectively. Three discrete Gaussian distributions over a 2-dimensional lattice L, with the same center $\mathbf{c}$ but three different standard deviations σ, are presented in Figure 4.1. In [GPV08], Gentry *et al.* showed that Klein's algorithm [Kle00] can be used to sample points according to a distribution statistically close to the desired discrete Gaussian distribution over any lattice L.[2]

After its introduction, Gaussian sampling became a *fundamental building block* in provable lattice-based cryptography. Alternative algorithms with smaller time complexity were proposed by Peikert and Micciancio [Pei10, MP12]. These general samplers over lattice were subsequently analyzed and improved by Ducas and Nguyen [DN12a] using floating point arithmetic and laziness techniques. They showed that "lazy techniques" can limit the need for high precision; one can use floating-point numbers at double precision (53 bits) most of the time, native on high-end architectures but still costly on embedded devices. The resulting algorithms run in quasi-quadratic time $\tilde{\mathcal{O}}(n^2)$ in the size of the input basis n. Following our work, Ducas described in his Ph.D. thesis,

[1] Some authors write a probability proportional to $\exp(-\pi\|\mathbf{x}-\mathbf{c}\|^2/s^2)$ where $s = \sqrt{2\pi}\sigma$.

[2] Note that producing an *exact* discrete Gaussian distribution is possible by autorizing the compuration time to be arbitrarily large [BLP+13, Section 5]. Here we focus in sampling from distributions that are *close* to discrete Gaussian distributions (using truncation); the required precision being driven by the security proof. In all this chapter, we focus in sampling from a distribution *statistically* close to the desired distribution (governed by our signature scheme in Chapter 6), *i.e.* that the statistical difference between the distributions is negligible.



(a) $\sigma = 1.3$ (b) $\sigma = 0.9$ (c) $\sigma = 0.5$

Figure 4.1 – Three discrete Gaussian distributions with support a 2-dimensional lattice and with the same center but different standard deviations σ. Note that the z-axis represents the probabilities of the elements to be outputted.

as on-going work, a variant of Klein's algorithm running in quasi-quadratic time without using floating point arithmetic nor high precision computations [Duc13].

One-Dimensional Gaussian Sampling. A fundamental building block of *all* these latter algorithms is the centered (in $\mathbf{0}$) one-dimensional discrete Gaussian sampling over the integers (*i.e.* $L = \mathbb{Z}$). This simple distribution is also the *only* used Gaussian distribution in Lyubashevsky's signature scheme [Lyu12] and in numerous fully homomorphic encryption schemes [BV11a, BV11b, Bra12, FV12, BLLN13]. Note that sampling according to a discrete Gaussian distribution of standard deviation σ over $L = \mathbb{Z}^n$ for $n \geqslant 1$ can be reduced to sampling n integers from the discrete Gaussian distribution of standard deviation σ over $\mathbb{Z}$ [DG14, Lemma 2].

Constrained Devices. In order to be widely deployed, standard cryptographic primitives such as signature schemes and encryption schemes have to be implemented on constrained devices (*e.g.* smart-cards). Constrained devices have a *limited memory storage* and are not adapted to high-precision computations nor floating-point arithmetic.

With the noticeable exception of NTRU [HPS98, HHGP$^+$03], lattice-based cryptosystems operating at a standard security level have remained out of reach of constrained devices by *several orders of magnitude*. Indeed, most of lattice-based cryptosystems admit large public keys (vector or matrices of several hundreds of elements) or large signatures. This already cripples any hope for being implemented on small architectures. Moreover, to be provably secure, these cryptosystems use discrete Gaussian sampling. In 2012, Dwarakanath and Galbraith made an extensive survey on discrete Gaussian sampling over $\mathbb{Z}$ with particular focus on constrained devices [DG14]. This paper concluded that none of the existing methods are particularly suited for these environments. Indeed, they require either a large memory storage or high-precision floating-point arithmetic, if not both.

A first step towards a practical lattice-based signature scheme was achieved by [GLP12] with an implementation on a low-cost FPGA, by *avoiding Gaussian distributions*, at the cost of some compactness and security compared to [Lyu12].

Our Contribution. In this chapter we focus on how to efficiently sample from a distribution statistically close to a one-dimensional discrete Gaussian distribution over the integers, *i.e.* over $\mathbb{Z}$, adapted for constrained devices. Our new algorithm does *not* require large memory storage nor high precision computation of transcendental functions. In particular, compared to known algorithms, we achieve an *exponential improvement* in the size of precomputed tables.

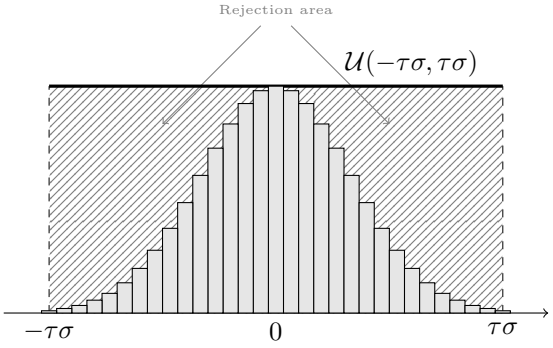


Figure 4.2 – Basic Rejection Sampling for Discrete Gaussian Distribution.

Outline. In Section 4.2, we review some known techniques to sample over $\mathbb{Z}$. In Section 4.3 we present an efficient method to sample according to a Bernoulli distribution with bias $\exp(-x/f)$ for a fixed $f \in \mathbb{R}$ from a small table of precomputed values and without actually computing transcendental function. In Section 4.4, we give a method to easily sample from an exponential distribution that is close to a discrete Gaussian, called binary discrete Gaussian distribution, and then use rejection sampling so that the output is statistically close to the desired Gaussian. Finally we conclude in Section 4.5.

4.2 Discrete Gaussian Sampling: Prior Art

There are two generic methods to sample according to a discrete Gaussian distribution D_σ centered in 0: rejection sampling and the inversion method [DG14].[3] First, note that it is generally useful to ignore large values which are unlikely to appear when drawing according to a Gaussian distribution.

Lemma 4.1 ([MR07])**.** *For any dimension m, $\sigma > 0$ and $\tau > 1$, $\rho_\sigma(\mathbb{Z}^m \setminus \tau\sigma\sqrt{m}\mathfrak{B}) < 2C(\tau)^m \cdot \rho_\sigma(\mathbb{Z})^m$, where $C(\tau) = \tau \exp\left(\frac{1-\tau^2}{2}\right) < 1$, and $\mathfrak{B}$ is the centered ℓ_2 unit ball.*

Therefore, to obtain a distribution $2^{-\lambda}$-close to a 1-dimensional Gaussian, one should choose the tailcut parameter $\tau \approx \sqrt{\lambda \cdot 2\ln 2}$, the typical value being $\tau = 12$ for $\lambda = 100$ (see [Lyu12, Lemma 4.4]).

Gaussian Sampling from Rejection Sampling. This method was proposed in [GPV08] and uses basic rejection sampling as follows: choose a uniform integer $x \in S := \{-\tau\sigma, \ldots, \tau\sigma\}$ and accept it with probability proportional to $\exp(-x^2/2\sigma^2)$; restart otherwise. A graphical representation is given in Figure 4.2. This algorithm requires about $2\tau/\sqrt{2\pi}$ trials in average, and thus $\mathcal{O}(\tau \log_2(\sigma))$ bits of entropy using laziness.[4] The main drawback is the need to compute the exponential function with very high-precision. Additionally, an average of $2\tau/\sqrt{2\pi} \approx 10$ trials until acceptance is rather expensive.

Gaussian Sampling from the Inversion Method. The inversion method, suggested in [Pei10], makes use of a cumulative distribution table to sample more efficiently (with complexity $O(\log_2 \sigma)$) and is very efficient when given enough memory. It uses the fact that, if U is a uniform random variable over $[0, 1]$, the random variable $D_\sigma^{-1}(U)$ has the desired distribution over $\mathbb{Z}$. One tabulates the approximate

[3] One could also sample according to a continuous Gaussian distribution, round it and use rejection sampling to obtain a discrete Gaussian distribution. However, sampling according to a continuous Gaussian distribution requires high precision, not to mention computations of transcendental functions such as $\exp, \log, \cos$ or $\sin$; we therefore do not consider it in this thesis.

[4] Laziness is an algorithmic trick saving both computation and entropy consumption; for our purpose, it is used in two cases of application. First, as in many compilers, when computing $a \wedge b$ and $a \vee b$, b is not always evaluated depending on the value of a. The second concerns the comparisons of reals of the form $r < c$: the result might be decided only knowing their first different bit; for a uniform $r \in [0, 1)$, only 2 bits are needed on average. In practice however, one may apply this technique word by word rather than bit by bit.

cumulative distribution of the desired distribution, *i.e.* the probabilities $p_z = \Pr[x \leqslant z : x \leftarrow D_\sigma]$ for $z \in S$, precomputed with λ bits of precision. At sampling time, one generates $u \in [0, 1)$ uniformly at random and perform a binary search through the table to locate some $z \in S$ such that $u \in [p_{z-1}, p_z)$ and outputs z. This approach consumes $\mathcal{O}(\log_2(\sigma))$ bits of entropy, which is optimal up to a constant factor.[5] In [DG14], Dwarakanath and Galbraith suggest to combine this method with the Knuth-Yao algorithm (this approach was later implemented in [RVV13]). This leads to a significant decrease of the table size by a factor slightly less than 2.

Remark 4.2. Sampling according to a continuous Gaussian distribution is widely used in traditional applications in statistical computing (*e.g.* signal processing, finance, ...) but cryptographic applications require higher quality sampling (*i.e.* with a smaller statistical distance between the desired distribution and the actual distribution sampled). Therefore, the discretization of continuous Gaussian sampling techniques (using rounding) does not present any advantage compared to the aforementioned techniques.

Limitations of the Known Algorithms. Unfortunately, neither of the two approaches is appropriate for use in constrained devices [DG14]. Indeed, using rejection sampling requires floating-point arithmetic while there is no floating-point hardware co-processor. Floating-point arithmetic is too costly to allow an on-card discrete Gaussian sampling via rejection sampling.

As the inversion method is concerned, the size of the look-up table is crippling any hope to fit on a constrained device. Indeed, let us consider the parameters of Chapter 6 for the signature scheme BLISS. In particular, one wants to sample according to a discrete Gaussian distribution with standard deviation $\sigma \geqslant 107$ for 128 bits of claimed security. Being conservative (*i.e.* taking $\tau = 12$ and $\sigma = 107$) and using the inversion method, one would need to store $\tau \cdot \sigma \cdot 128 \geqslant 20$kB of precomputed values. In other words, at least 4% of the *whole smart-card capacity* would be used just to store values used during discrete Gaussian sampling (which is a building block of cryptographic primitives).[6] Storing 20kB of precomputed data might be potentially acceptable for some devices and applications, but is completely impractical in most cases (memory is really expensive on smart-cards). Note also that another drawback of the inversion method is the high amount of memory access that slow down the implementation (see also Section 4.5).

In the rest of this chapter, we describe a discrete Gaussian sampler using exponentially less memory and no floating-point arithmetic, therefore adapted to constrained environments.

4.3 Efficient Sampling from Bernoulli Distributions

We recall that a Bernoulli distribution $\mathcal{B}_a$ assigns 1 (True) with probability $a \in [0, 1]$ and 0 (False) with probability $1 - a$. Overloading the notation for the sake of clarity, we will denote by $\mathcal{B}_a$ both the distribution and a generic random variable that follows that distribution independently of all others. In particular, we have for any $a, b \in [0, 1]$ that $\neg\mathcal{B}_a = \mathcal{B}_{1-a}$, $\mathcal{B}_a \vee \mathcal{B}_b = \mathcal{B}_{a+b-ab}$, $\mathcal{B}_a \oplus \mathcal{B}_b = \mathcal{B}_{a+b-2ab}$ and

$$\mathcal{B}_a \wedge \mathcal{B}_b = \mathcal{B}_{ab} \ . \tag{4.1}$$

Before describing our technique to sample from a Bernoulli distribution with an exponential bias computed on the fly, recall that sampling from a distribution statistically close (*i.e.* $(2^{-\lambda})$-close) to a Bernoulli distribution $\mathcal{B}_a$ for a given bias a is easy: take an approximation of a up to λ correct bits, then sample a uniform real $r \in [0, 1)$ up to λ bits of precision and answer 1 if and only if $r < a$. (As previously recalled, on average one only needs to compute 2 bits of r.)

Bernoulli Distribution with Exponential Bias. Our problem is as follows: For a fixed real f, a positive integer $x \leqslant 2^\ell$ given as input, sample a random boolean according to $\mathcal{B}_{\exp(-x/f)}$. Combining the simple homomorphic property of the exponential function with Equation (4.1), our approach, implemented by Algorithm 4.1, requires only ℓ precomputed entries with λ bits of precision, and *no* evaluation of transcendental functions.

[5] Indeed, the entropy of the discrete Gaussian distribution of standard deviation σ over $\mathbb{Z}$ is $1.4 + \log_2(\sigma)$ bits [DG14].

[6] Some numbers to keep in mind for a (standard) smart-card are the following: a processor of 20Mhz with a 32 bit CPU, 20kB of RAM, 32kB of ROM (secure boot, ...) and 500kB for all the program/data (flash).

Algorithm 4.1 Sampling $\mathcal{B}_{\exp(-x/f)}$ for $x \in [0, 2^\ell)$ using precomputed values $\{a_i = \exp(-2^i/f)\}_{i=0,\dots,\ell-1}$.

```
1: function BernoulliExp_f(x = Σ_{i=0}^{ℓ} x_i 2^i ∈ [0, 2^ℓ))
2:     for i = ℓ − 1 downto 0 do
3:         if x_i then
4:             A_i ← B_{a_i}
5:             if ¬A_i then
6:                 return 0                ▷ Laziness: output 0 when a sampled bit is equal to 0
7:             end if
8:         end if
9:     end for
10:    return 1                            ▷ Output 1 with probability exp(−x/f)
11: end function
```

Lemma 4.3. *For any integer $x \in [0, 2^\ell)$, Algorithm 4.1 outputs a bit according to $\mathcal{B}_{\exp(-x/f)}$.*

Proof. Denoting the binary decomposition of x by $x = \sum_{i=0}^{\ell-1} x_i 2^i$ with $x_i \in \{0, 1\}$, we have

$$\mathcal{B}_{\exp(-x/f)} = \mathcal{B}_{\exp(-\sum_i x_i 2^i/f)} = \mathcal{B}_{\prod_i \exp(-x_i 2^i/f)} = \bigwedge_{i \text{ s.t. } x_i=1} \mathcal{B}_{\exp(-2^i/f)} \,. \qquad \square$$

Remark 4.4. Note that Algorithm 4.1 is defined so that the smallest probabilities are checked first, so that the algorithm can terminate faster. Moreover this algorithm is very fast, and uses about ℓ bits of entropy on average for random x.

However this laziness technique (and the fact that we draw random bits only when $x_i = 1$) when using Algorithm 4.1 might leak information that could be exploited by an adversary. It might therefore be preferable to modify Algorithm 4.1 to do a constant time sampling, at the cost of the entropy consumption (see also Section 4.5).

Bernoulli Distribution with Inverse Hyperbolic Cosine Biases. During the implementation of our signature scheme [DDLL13a] built upon the discrete Gaussian sampling technique described in this chapter, one needs to reject with probability $1/\cosh(x/f)$ for a given f. As it would be a pity to remove the high precision computations from discrete Gaussian sampling but keep it later in our signing process, we developed a sampling algorithm for Bernoulli variables with inverse hyperbolic cosine biases.

Our problem is as follows: For a fixed real f, a positive integer $x \leqslant 2^\ell$ given as input, sample a random Boolean according to $\mathcal{B}_{1/\cosh(x/f)}$. Recall that

$$\frac{1}{\cosh(x/f)} = \frac{2}{\exp(|x|/f) + \exp(-|x|/f)} = \frac{\exp(-|x|/f)}{1/2 + 1/2 \cdot \exp(-2|x|/f)} \,. \tag{4.2}$$

To sample efficiently according to the Bernoulli distribution $\mathcal{B}_{1/\cosh(x/f)}$, we reuse the previous generator for $\mathcal{B}_{\exp(-|x|/f)}$ with no explicit evaluation of exp or cosh. In order to deal with the fraction in Equation (4.2), we introduce a new operation denoted $\oslash$ and computed according to Algorithm 4.2.

Lemma 4.5 (Correctness and Efficiency of Algorithm 4.2)**.** *For any $a, b \in (0, 1]$ we have, $\mathcal{B}_a \oslash \mathcal{B}_b = \mathcal{B}_{a/(1-(1-a)b)}$ and Algorithm 4.2 terminates after an average of $1/(1-(1-a)b)$ trials.*

Proof. At each trial, the probability of restarting is $(1-a)b$. Now, the probability that it outputs 1 is easily computed as the sum over each trial:

$$\Pr[\mathcal{B}_a \oslash \mathcal{B}_b = 1] = a \sum_{k=0}^{\infty} (1-a)^k b^k = \frac{a}{1-(1-a)b} \,. \qquad \square$$

Algorithm 4.2 Sampling $\mathcal{B}_a \oslash \mathcal{B}_b$.

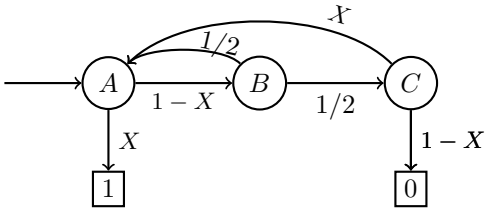
```
1: function BernoulliSlash(a, b)
2:     while True do
3:         A ← B_a
4:         if A then
5:             return 1                          ▷ output 1 when A = 1
6:         end if
7:         B ← B_b
8:         if ¬B then
9:             return 0                          ▷ output 0 when A = B = 0
10:        end if
11:    end while                                 ▷ restart otherwise
12: end function
```

Corollary 4.6. *Let $f \in \mathbb{R}$. For any $x \in \mathbb{R}$, we have*

$$\mathcal{B}_{1/\cosh(x/f)} = \mathcal{B}_{\exp(-|x|/f)} \oslash \left(\mathcal{B}_{1/2} \vee \mathcal{B}_{\exp(-|x|/f)}\right)$$

and Algorithm 4.2 requires less than 3 calls to $\mathcal{B}_{\exp(-|x|/f)}$ on average.

Proof. Correctness is a direct application of the previous lemma. Set $X = \exp(-|x|/f)$. Algorithm 4.2 for the computation of the Bernoulli variable $\mathcal{B}_X \oslash \left(\mathcal{B}_{1/2} \vee \mathcal{B}_X\right)$ can be seen as the following Markov chain:



Let $\mathbf{M}$ denote the restriction of the transition matrix to the states A, B and C (indexed in that order), and let $\mathbf{v} = (1, 0, 0)^t$ be the initial density vector. The density vector after k steps is $\mathbf{M}^k \cdot \mathbf{v}$, so the average number of steps through each state A, B and C is given by the vector

$$\mathbf{w} = (w_A, w_B, w_C)^t = \sum_{k=0}^{\infty} \mathbf{M}^k \cdot \mathbf{v} = (\mathbf{I}_3 - \mathbf{M})^{-1} \cdot \mathbf{v}$$

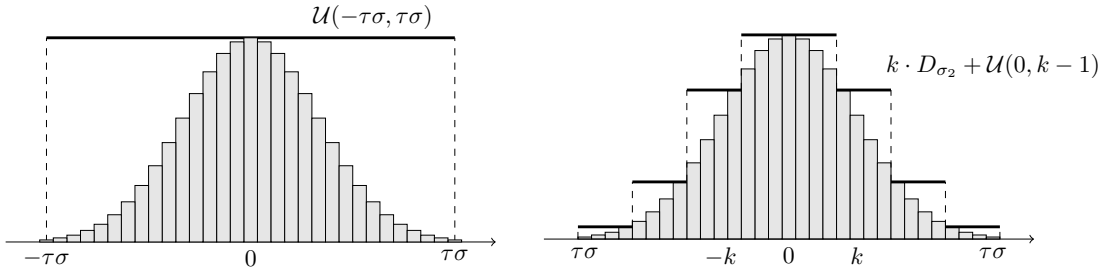
where

$$\mathbf{M} = \begin{pmatrix} 0 & \frac{1}{2} & X \\ 1-X & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{pmatrix} \quad \text{and} \quad (\mathbf{I}_3 - \mathbf{M})^{-1} \cdot \mathbf{v} = \frac{1}{1+X^2} \begin{pmatrix} 2 \\ -2X+2 \\ 1-X \end{pmatrix}.$$

Since the calls to $\mathcal{B}_{\exp(-|x|/f)}$ are performed during the states A and C, the average number of calls to this Bernoulli sampling is $C(X) := w_A + w_C = \frac{3-X}{1+X^2}$. Finally, we have $C(X) \leqslant 3$ for all $X > 0$. □

4.4 Reduce the Rejection Rate with a Binary Discrete Gaussian Distribution

Based on Algorithm 4.1 to sample efficiently from $\mathcal{B}_{\exp(-x/f)}$ for $x \geqslant 0$, it is now possible to obtain a Gaussian distribution via generic rejection sampling algorithm as in [GPV08], trading high-precision evaluation of transcendental functions against a table of $\log_2(\tau^2\sigma^2)$ precomputed values. Indeed, the method is adapted as follows. First, one stores in a table the values $\exp(-2^i/(2\sigma^2))$ with λ bits of precision for $i = 0, \ldots, \log_2(\tau^2\sigma^2)$. At sampling time, one chooses a uniform integer $x \in S := \{-\tau\sigma, \ldots, \tau\sigma\}$ and uses Algorithm 4.1 to generate a boolean according to $\exp(-x^2/(2\sigma^2))$; accept if the boolean is True and restart otherwise.



(a) from uniform distribution (repetition rate ≈ 10) (b) from our adapted distribution (repetition rate ≈ 1.47)

Figure 4.3 – Rejection Sampling.

Algorithm 4.3 Sampling $D^+_{\sigma_2}$.

```
1: function GaussianBinaryPositive( )
2:     b ← B_{1/2}
3:     if ¬b then
4:         return 0                          ▷ return 0 with probability 1/2
5:     end if
6:     for i = 1 to ∞ do
7:         b_1, ..., b_k ← B_{1/2} for k = 2i − 1
8:         if ∃j ∈ {1, ..., k − 1} such that b_j then
9:             restart                       ▷ restart if one of the k − 1 first bits is 1
10:        end if
11:        if ¬b_k then
12:            return i                      ▷ output i if the last bit is 0
13:        end if
14:    end for
15: end function
```

However, the algorithm still requires $(2\tau/\sqrt{2\pi}) \approx 10$ trials on average to output an x statistically close to the correct distribution. This is due to the significant distance between the uniform distribution and the target distribution. To solve this issue we introduce a new sampling algorithm with an average number of rejections smaller than 1.47. We achieve that result by sampling from a specific distribution denoted $\mathcal{D}_{k,\sigma_2}$, for which sampling is easy. The distribution $\mathcal{D}_{k,\sigma_2}$ is much closer to the target distribution $D_{k\sigma_2}$ than the uniform distribution (see Figure 4.3b versus Figure 4.3a), leading to a huge acceleration of rejection sampling.

The Binary Discrete Gaussian Distribution. Let us introduce the binary discrete Gaussian distribution D_{σ_2}, which is a discrete Gaussian with specific variance $\sigma_2 = \sqrt{1/(2\ln 2)} \approx 0.849$ and probability density

$$\rho_{\sigma_2}(x) = e^{-x^2/(2\sigma_2^2)} = 2^{-x^2} \quad \text{for } x \in \mathbb{Z}\,.$$

We will combine D_{σ_2} with the uniform distribution to produce the distribution $\mathcal{D}_{k,\sigma_2}$ (see Figure 4.3b). We will only focus on the positive half of D_{σ_2} denoted $D^+_{\sigma_2} = \{x \leftarrow D_{\sigma_2} : x \geqslant 0\}$. Algorithm 4.3 is designed to sample according to $D^+_{\sigma_2}$ very efficiently using only unbiased random bits.

Lemma 4.7. *Algorithm 4.3 outputs positive integers according to $D^+_{\sigma_2}$. On average, the algorithm terminates after $2/\rho_{\sigma_2}(\mathbb{Z}^+) < 1.3$ trials and consumes 2.6 bits of entropy.*

Proof. The probability that the algorithm returns $x \in \mathbb{Z}^+$ is $\rho_{\sigma_2}(x)/\rho_{\sigma_2}(\mathbb{Z}^+)$ where $\rho_{\sigma_2}(\mathbb{Z}^+) = \sum_{i=0}^{\infty} 2^{-i^2} \approx 1.564$. We now observe that the binary expansion of $\rho_{\sigma_2}(\{0,\ldots,j\}) = \sum_{i=0}^{j} 2^{-i^2}$ is of

Algorithm 4.4 Sampling $D^+_{k\sigma_2}$ for $k \in \mathbb{Z}^+$.

```
1: function GaussianPositive(k)
2:     x ← GaussianBinaryPositive()
3:     y ← {0, ..., k − 1}
4:     z = xk + y
5:     b ← BernoulliExp_{2(kσ2)^2}(−y(y + 2xk))
6:     if ¬b then
7:         restart                    ▷ restart with probability exp(−y(y + 2xk)/2(kσ2)^2)
8:     end if
9:     return z
10: end function
```

the form

$$\rho_{\sigma_2}(\{0,\dots,j\}) = 1\,.\,1\,0\,0\,1\underbrace{0\dots0}_{4}1\underbrace{0\dots0}_{6}1 \quad \dots \quad \underbrace{0\dots0}_{2(j-2)}1\underbrace{0\dots0}_{2(j-1)}1\,.$$

Thus, each trial of Algorithm 4.3 implicitly chooses a random real $r \in [0,2)$ that will be rejected if $r > \rho_{\sigma_2}(\mathbb{Z}^+)$. It then computes the cumulative table (scaled by $\rho_{\sigma_2}(\mathbb{Z}^+)$) on the fly and reject if necessary. On average, the algorithm completes after $2/\rho_{\sigma_2}(\mathbb{Z}^+) < 1.3$ trials and consumes 2.6 bits of entropy. □

Building the Centered Discrete Gaussian Distribution. Based on our efficient sampling for the distribution $D^+_{\sigma_2}$, we can now easily build the positive discrete Gaussian distribution with standard deviation $\sigma = k\sigma_2$ for $k \in \mathbb{Z}^+$. We refer to our Algorithm 4.4 based on the property

$$\mathcal{D}^+_{k,\sigma_2} = k \cdot D^+_{\sigma_2} + \mathcal{U}(0, k-1),$$

and where we reject the result with probability $\exp(-y(y+2kx)/(2\sigma^2))$ where x and y respectively follow the distributions $D^+_{\sigma_2}$ and $\mathcal{U}(0,k-1)$, the uniform distribution over $[0,k-1] \cap \mathbb{Z}$.

Theorem 4.8. *For any non-negative integer input k, Algorithm 4.4 outputs positive integers according to D^+_σ for $\sigma = k\sigma_2$. On average, it requires less than 1.47 trials. Consequently, Algorithm 4.5 outputs integers according to D_σ, and requires about $1 + \frac{1}{5\sigma}$ trials.*

Remark 4.9. Entropy consumption for each trial of Algorithm 4.4 is: 2.6 bits for $x \leftarrow D^+_{\sigma_2}$, $\log_2 k$ bits for $y \leftarrow \mathcal{U}(0,k-1)$, and $\approx 1 + \log_2 \sigma$ bits (measured in practice) for the rejection bit $b \leftarrow \mathcal{B}_{\exp(-y(y+2kx)/(2\sigma^2))}$. Since on average, Algorithm 4.4 restarts 1.47 times, we an average entropy consumption of $\approx 7 + 3\log_2 \sigma$ when using Algorithm 4.5.

Proof. Let us start with the fact that any output z is uniquely written as $kx+y$ for $y \in \{0,\dots,k-1\}$. The input (resp. desired output) distribution weight function g (resp. f) is

$$g(z) = g(kx+y) = \frac{\rho_{\sigma_2}(x)}{k\rho_{\sigma_2}(\mathbb{Z}^+)} \quad \text{and} \quad f(z) = f(kx+y) = \frac{\rho_{k\sigma_2}(kx+y)}{\rho_{k\sigma_2}(\mathbb{Z}^+)}\,.$$

Since we restrict the distribution to non-negative integers ($x, y \geqslant 0$), we have $\exp\left(-\frac{y(y+2kx)}{2\sigma^2}\right) \leqslant 1$. Therefore, the probability to output some integer $z = kx + y$ is proportional to

$$\rho_{\sigma_2}(x)\exp\left(-\frac{y(y+2kx)}{2\sigma^2}\right) = \exp\left(-\frac{x^2}{2\sigma_2^2} - \frac{2kxy+y^2}{2\sigma^2}\right) = \exp\left(-\frac{(kx+y)^2}{2\sigma^2}\right) = \rho_{k\sigma_2}(z)\,.$$

The repetition rate M is upper-bounded by

$$M = \max\frac{f}{g} \leqslant \frac{k\rho_{\sigma_2}(\mathbb{Z}^+)}{\rho_{k\sigma_2}(\mathbb{Z}^+)} \leqslant \frac{k\rho_{\sigma_2}(\mathbb{Z}^+)}{k\sigma_2\sqrt{\pi/2}} \leqslant 1.47\,,$$

Algorithm 4.5 Sampling $D_{k\sigma_2}$ for $k \in \mathbb{Z}^+$.

1: **function** GAUSSIAN(k)
2: $z \leftarrow$ GAUSSIANPOSITIVE(k)
3: $b \leftarrow \mathcal{B}_{1/2}$
4: **if** $z = 0$ and $\neg b$ **then**
5: **restart** ▷ restart with probability 1/2 if $z = 0$
6: **end if**
7: **return** $(-1)^b \cdot z$
8: **end function**

Table 4.1 – Comparison of Discrete Gaussian Sampling Algorithms over the Integers.

	No FPA	**Precomputation Storage**	**Table Look-ups**	**Entropy Consumption**
Naive rejection [GPV08]	$\times$ $(2\tau/\sqrt{2\pi})$	0	0	$2\tau/\sqrt{2\pi} \cdot \log_2(\tau\sigma)$
Naive rejection with precomputed table	✓	$\lambda \cdot \log_2(\tau^2\sigma^2)$	$2\tau/\sqrt{2\pi}$	$2\tau/\sqrt{2\pi} \cdot \log_2(\tau\sigma)$
Cumulative Distribution Table [Pei10]	✓	$\lambda \cdot \tau\sigma$	$\log_2(\tau\sigma)$	$\approx 2 + \log_2 \sigma$ (*cf.* [DG14])
Our Method (Algorithm 4.5)	✓	$\lambda \cdot \log_2(2.4\tau\sigma^2)$	$1.5 \log_2(\sigma)$	$\approx 7 + 3 \log_2 \sigma$ (*cf.* Remark 4.9)

where the second inequality follows from the sum-integral comparison ($\rho_{k\sigma_2}$ is decreasing over $[0, \infty)$)

$$\rho_{k\sigma_2}(\mathbb{Z}^+) \geqslant \int_{x=0}^{\infty} \rho_{k\sigma_2}(x)dx = k\sigma_2\sqrt{\pi/2}\,. \qquad \square$$

Finally, we apply Algorithm 4.5 to build the (full) discrete Gaussian distribution D_σ over $\mathbb{Z}$.

Remark 4.10. Note that our discrete Gaussian sampling *requires* $\sigma \in \sigma_2\mathbb{Z}^+$. However, in lattice-based cryptography, security requirements provide a lower bound on σ: it suffices then to sample according to a discrete Gaussian distribution of standard deviation $\lceil \sigma/\sigma_2 \rceil \sigma_2$.

Remark 4.11. Note that Algorithm 4.5 to sample from D_σ can easily be adapted to sample from $D_{\sigma,c}$ for $c \in \mathbb{Z}$, by simply adding c to the result. Sampling from $D_{\sigma,c}$ for $c \in \mathbb{R}$ with less than 1.6 trials on average is also possible at the cost of doubling the memory requirement [Duc13, Section 7.1.4]. The main idea is to sample from a wider distribution $D_{\sigma'}$ ($\sigma' = 5\sigma/4$ in [Duc13]) and reject using BERNOULLIEXP$_f(x)$ as in Line 5 of Algorithm 4.4 for well chosen x, f.

4.5 Conclusion

In this chapter, we proposed very simple and natural algorithm to sample from a distribution statistically close to a discrete Gaussian distribution over the integers when the standard deviation is in $\sqrt{1/(2\log 2)}\mathbb{Z}^+$. Our algorithm requires exponentially less memory than the usual inversion method ($\lambda \log_2(2.4\tau\sigma^2)$ bits versus $\lambda\tau\sigma$ bits) and consumes only 3 times more entropy (*cf.* Table 4.1). For example, using the parameters of Chapter 6 for the signature scheme BLISS (*i.e.* $\sigma = 107$ for $\lambda = 128$), our algorithm requires 293 Bytes of memory versus 20 kBytes for the inversion method; such storage requirement is adapted to constrained devices.

A proof-of-concept implementation of our algorithms is available under the free software license CeCILL[7] at [DL13]. We also provided an implementation of the inversion method for the sake of comparison.

[7] `http://www.cecill.info/`

Table 4.2 – Comparison of Discrete Gaussian Sampler Algorithms over the Integers ($\sigma = 215$ and ≈ 1640 Runs).

Algorithms	Cycles	Memory	No FPA
Ziggurat (optimized for speed)	1,014,253	10240B	×
Ziggurat (average)	1,761,321	5120B	×
Ziggurat (optimized for size)	3,776,097	2560B	×
Knuth-Yao	1,233,918	19064B	✓
Bernoulli (our algorithm)	934,131	900B	✓

Practical Implementations of Discrete Gaussian Sampling Algorithms. Concurrently to our work, two implementations of discrete Gaussian sampling have been proposed at SAC 2013 [RVV13, BCG$^+$13].

The Knuth-Yao algorithm described in [DG14] was implemented in [RVV13] on FPGAs. This algorithm constructs a binary tree of the probabilities $p_x = \rho_\sigma(x)/\rho_\sigma(\mathbb{Z}^+)$ for $x \in \mathbb{Z}^+$. To sample from an uniform distribution, one walks down the tree from the root using one uniform bit at each step to decide which of the two children to move to. When one hits a leaf, one outputs the integer label x for this leaf. Unfortunately the main drawback of this algorithm is the storage of the tree and the high amount of memory access needed.

In [BCG$^+$13], the authors adapted the continuous Ziggurat algorithm to sample from a discrete Gaussian distribution and offer a flexible time-memory trade-off. The Ziggurat algorithm seems quite similar (when depicted) to our algorithm. Precomputed rectangles are used to sample x. One creates m rectangles with the left corners on the y-axis and the right corners on the graph of the probability distribution function such that all rectangles have the same size. The entire area under the graph is then covered by rectangles and a rectangle R_i can efficiently be stored by just storing the coordinates (x_i, y_i) of the lower right corner. Then one performs the basic rejection sampling step (using floating-point arithmetic) using this new distribution. The main drawback of this method is therefore the requirement for either floating-point arithmetic (with few rectangles) to optimize the memory, or large precomputed tables (with a lot of rectangles) to optimize the speed, or a combination thereof.

In [OPG14], Oder, Pöppelmann and Güneysu investigated the latter potential approaches (our algorithm [DDLL13a], the Ziggurat method [BCG$^+$13] and the Knuth-Yao algorithm [RVV13]) to sample from a discrete Gaussian distribution on an ARM Cortex-M4F micro-controller.[8] They conclude that our sampler is the *best choice* on this constrained device compared to the other samplers, as illustrated by Table 4.2. This work also illustrates that on constrained devices, contrary to our naive implementation [DL13], the Knuth-Yao algorithm with the larger table and (nearly) optimal entropy consumption does not outperform our Gaussian sampling algorithm because of the high amount of memory access.

Future Work and Perspectives. An important future work is to evaluate the resistance of discrete Gaussian sampling algorithms against side-channel attacks. In particular, all previous algorithms are not constant time and therefore might leak useful information for an adversary. Assessing whether constant time sampling is necessary, or how to do it efficiently is therefore a promising work. It also seems interesting to consider software-hardware co-design to implement lattice-based schemes, where *e.g.* the hardware co-processor could be dedicated to perform discrete Gaussian sampling. Indeed, when used as a building block of lattice-based signature schemes, discrete Gaussian sampling takes more than 35% of the running-time [DDLL13a, BB13].

[8] Note that this device is more efficient and has more memory than a smart-card. It remains an open problem to perform discrete Gaussian sampling over a smart-card or an RFID tag.

Chapter 5

Design of BLISS, an Efficient Lattice-Based Signature Scheme

5.1 Introduction

This chapter proposes a construction of a lattice-based signature scheme that improves over today's most efficient lattice-based schemes. The heart of the improvement consists in a modification of the rejection sampling algorithm of Lyubashevsky's signature scheme [Lyu12] (and of several other lattice primitives [Lyu09, Rüc10, DPSZ12]). Our new rejection sampling algorithm which samples from a *bimodal* Gaussian distribution, combined with a modified scheme instantiation, ends up reducing the standard deviation of the resulting signatures by a factor that is asymptotically square root in the security parameter. Practical instantiations of our signature scheme for security levels of 128, 160 and 192 bits are given in Chapter 6 and compare very favorably to existing schemes such as RSA and ECDSA in terms of efficiency. In addition, our instantiation has shorter signature and public key sizes than all previously proposed lattice-based signature schemes.

This chapter was part of the article Lattice Signature and Bimodal Gaussian [DDLL13a], cosigned with L. Ducas, A. Durmus and V. Lyubashevsky and published at Crypto 2013 [CG13a]. The full version of the article is available at [DDLL13b].

Background. Early lattice-based signature schemes proposals GGH and NTRUSign are directly related to the difficulty to solve a certain lattice problem, but lack a security proof (*cf.* [GGH97, HHGP+03]). It was known from the beginning that each signature was leaking information on the signer's secret key but no attack exploiting this leakage was known. Still, some heuristic countermeasures were proposed to make this statistical leakage unusable [HHGP+03]. In 2006, Nguyen and Regev [NR09] presented the first successful key-recovery experiments on GGH and the raw version of NTRUSign (with as few as 400 signatures). The latter attack can be adapted [MPSW09, DN12b] to break all the heuristic countermeasures designed to patch NTRUSign, and in particular, to the standardized version of NTRUSign [IEE08].

Provably-secure lattice-based schemes are either based on the hash-and-sign paradigm [GPV08, MP12] or on Fiat-Shamir paradigm [Lyu09, Lyu12] and rejection sampling (Lemma 3.5). Quite surprisingly, the hash-and-sign schemes [MP12, BB13] seem significantly less efficient than the Fiat-Shamir schemes [Lyu12, GLP12, GOPS13]. The most efficient instantiations have both signature and key size of the order of 9kb [GLP12, GOPS13] for approximately 80 bits of security.[1]

At the heart of Lyubashevsky's signature schemes [Lyu09, Lyu12] is rejection sampling. Its first use in lattice-based constructions was presented in [Lyu08] to construct a three-round identification scheme. A standard identification scheme is a three round sigma protocol that consists of commit, challenge, and response stages. Unfortunately, lattice-based constructions of such an identification scheme are more intricate than for number-theoretic schemes.

[1] In [GLP12], a 100-bit security level was claimed, but the cryptanalysis of Chapter 6, which combines lattice-reduction attacks with combinatorial meet-in-the-middle techniques, estimates the actual security to be around 75-80 bits.

For example, let us describe Schnorr identification scheme [Sch89]. Let p be a large prime such that the discrete logarithm problem in $\mathbb{Z}_p$ is intractable and let q be a large prime divisor of $p-1$. Let s be uniformly generated in $\mathbb{Z}_q$ and set $\mathsf{sk} = s$. Let g be a generator of a subgroup of $\mathbb{Z}_p$ or order q, define $S = g^s$ and set $\mathsf{pk} = (q, g, S)$. In Schnorr identification scheme, a prover P wants to prove to a verifier V the knowledge of the secret key $\mathsf{sk} = s$ corresponding to the public key pk without revealing any information on s. The identification protocol is as follows.

Prover P		Verifier V
$y \leftarrow \mathbb{Z}_q$		
$Y = g^y$	$\xrightarrow{\quad Y \quad}$	
	$\xleftarrow{\quad c \quad}$	$c \leftarrow \mathbb{Z}_q$
$z = y + cs \bmod q$	$\xrightarrow{\quad z \quad}$	If $g^z = YS^c$ then output 1, else output 0.

The main idea in Schnorr scheme (and also GQ schemes [BP02]) is that the value y committed at the first stage is randomly and uniformly selected to hide the secret key s in the third stage. Indeed, since all operations are performed in $\mathbb{Z}_p$ – a finite ring –, a uniformly random y completely hides s. In lattice-based constructions however, security reductions require y to be chosen *small*. Therefore adding a challenge-dependent function of s is susceptible to leak some useful information on s. Thus, rejection sampling is performed so that s is not leaked when we add y to it (we describe this idea in much greater detail below). Improvements in lattice-based identification schemes (and therefore signature schemes via the Fiat-Shamir transformation) partly came via picking distributions that were more amenable to rejection sampling (*cf.* Figure 3.2b).

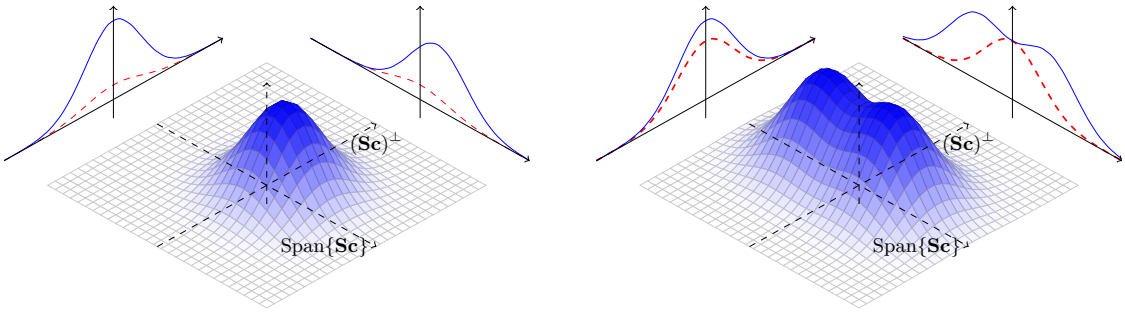
Lattice-Based Signature Scheme from [Lyu12]. Let us describe how the simplest version based on SIS of this scheme works. The secret key is an $m \times n$ matrix $\mathbf{S}$ with small coefficients, and the public key consists of a random $n \times m$ matrix $\mathbf{A}$ whose entries are uniform in $\mathbb{Z}_q$ and $\mathbf{T} = \mathbf{AS} \bmod q$. There is also a cryptographic hash function H, modeled as a random oracle, which outputs elements in $\mathbb{Z}^n$ with small norms. To sign a message digest μ, the signing algorithm first picks a vector $\mathbf{y}$ according to the distribution D_σ^m, where D_σ^m is the discrete Gaussian distribution over $\mathbb{Z}^m$ with standard deviation σ (*cf.* Chapter 4). The signer then computes $\mathbf{c} = H(\mathbf{Ay} \bmod q, \mu)$ and produces a potential signature $(\mathbf{z}, \mathbf{c})$ where $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$. Note that the distribution of $\mathbf{z}$ depends on the distribution of $\mathbf{Sc}$, and thus on the distribution of $\mathbf{S}$ – in fact, the distribution of $\mathbf{z}$ is exactly D_σ^m *shifted* by the vector $\mathbf{Sc}$.

To remove the dependence of the signature on $\mathbf{S}$, rejection sampling is used. The target distribution that we want for signatures is D_σ^m, whereas we obtain samples from the distribution D_σ^m shifted by $\mathbf{Sc}$ (call this distribution $D_{\mathbf{Sc},\sigma}^m$). To use rejection sampling, we need to find a positive real M such that for all (or all but a negligible fraction) $\mathbf{x}$ distributed according to D_σ^m we have $D_\sigma^m(\mathbf{x}) \leqslant M \cdot D_{\mathbf{Sc},\sigma}^m(\mathbf{x})$. A simple calculation (see [Lyu12, Lemma 4.5]) shows that

$$D_\sigma^m(\mathbf{x}) / D_{\mathbf{Sc},\sigma}^m(\mathbf{x}) = \exp\left(\frac{-2\langle \mathbf{x}, \mathbf{Sc}\rangle + \|\mathbf{Sc}\|^2}{2\sigma^2}\right). \tag{5.1}$$

The value of $\langle \mathbf{x}, \mathbf{Sc}\rangle$ behaves in many ways as a one-dimensional discrete Gaussian, and it can be thus shown that $|\langle \mathbf{x}, \mathbf{Sc}\rangle| < \tau\sigma\|\mathbf{Sc}\|$ with probability $1 - \exp(-\Omega(\tau^2))$. Asymptotically, the value of τ is proportional to the square root of the security parameter. Concretely, if we would like to have, for example, $1 - 2^{-100}$ certainty that $|\langle \mathbf{x}, \mathbf{Sc}\rangle| < \tau\sigma\|\mathbf{Sc}\|$, we would set $\tau = 12$. Thus with probability $1 - \exp(-\Omega(\tau^2))$, we have $\exp\left(\frac{-2\langle \mathbf{x},\mathbf{Sc}\rangle + \|\mathbf{Sc}\|^2}{2\sigma^2}\right) \leqslant \exp\left(\frac{2\tau\sigma\|\mathbf{Sc}\| + \|\mathbf{Sc}\|^2}{2\sigma^2}\right)$. So if $\sigma = \tau\|\mathbf{Sc}\|$, we will have $D_\sigma^m(\mathbf{x}) / D_{\mathbf{Sc},\sigma}^m(\mathbf{x}) \leqslant \exp\left(1 + \frac{1}{2\tau^2}\right)$. Therefore if we set $M = \exp\left(1 + \frac{1}{2\tau^2}\right)$, rejection sampling outputs signatures that are distributed according to D_σ^m where $\sigma = \tau\|\mathbf{Sc}\|$ and the expected number of repetitions is $M \approx \exp(1)$.[2] Upon receiving the signature $(\mathbf{z}, \mathbf{c})$ of μ, the verifier checks that $\|\mathbf{z}\|$ is "small" (roughly $\sigma\sqrt{m}$) and also that $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc} \bmod q, \mu)$. It is easy to check that the outputs of the signing procedure satisfy the two requirements.

[2] More precisely $\sigma = \tau \max_{\mathbf{S},\mathbf{c}} \|\mathbf{Sc}\|$, since $\mathbf{Sc}$ is not known in advance.



(a) In the original scheme of [Lyu12]

(b) In our scheme

Figure 5.1 – Improvement of Rejection Sampling with Bimodal Gaussian Distributions. In blue is the distribution of $\mathbf{z}$, for fixed $\mathbf{Sc}$ and over the space of all $\mathbf{y}$ in Figure (a) and all $(b, \mathbf{y})$ in Figure (b), before the rejection step and its decomposition as a Cartesian product over $\mathrm{Span}\{\mathbf{Sc}\}$ and $(\mathbf{Sc})^{\perp}$. In dashed red is the target distribution scaled by $1/M$.

Our Results and Techniques. In this work, we show how to remove the factor τ (in fact even more) from the required standard deviation. Above, we described how to perform rejection sampling when we were sampling potential signatures as $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$. Consider now, an alternative procedure, where we first uniformly sample a bit $b \in \{0, 1\}$ and then choose the potential signature to be $\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{Sc}$. In particular $\mathbf{z}$ is now sampled from the distribution $\frac{1}{2}D^m_{\mathbf{Sc},\sigma} + \frac{1}{2}D^m_{-\mathbf{Sc},\sigma}$. Assuming our target distribution is still D^m_σ, then we need to have, as above, $D^m_\sigma(\mathbf{x}) / \left(\frac{1}{2}D^m_{\mathbf{Sc},\sigma}(\mathbf{x}) + \frac{1}{2}D^m_{-\mathbf{Sc},\sigma}(\mathbf{x})\right) \leqslant M$. By using Equation (5.1) and some algebraic manipulations (see Section 5.3.2), we obtain that

$$\begin{aligned} D^m_\sigma(\mathbf{x}) / \left(\frac{1}{2}D^m_{\mathbf{Sc},\sigma}(\mathbf{x}) + \frac{1}{2}D^m_{-\mathbf{Sc},\sigma}(\mathbf{x})\right) &= \exp\left(\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) / \cosh\left(\frac{\langle \mathbf{x}, \mathbf{Sc}\rangle}{\sigma^2}\right) \\ &\leqslant \exp\left(\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right), \end{aligned}$$

where the last inequality follows from the fact that $\cosh(y) \geqslant 1$ for all y. Thus for rejection sampling to work with $M = \exp(1)$, as in the previous example, we only require that $\sigma = \|\mathbf{Sc}\|/\sqrt{2}$ rather than $\tau\|\mathbf{Sc}\|$.[3]

Our improvement is depicted in Figure 5.1, where Figure 5.1a shows the rejection sampling as done in [Lyu12]. There, the distribution D^m_σ (the dashed red line) must be scaled down by a somewhat large factor so that all but a negligible fraction of it fits under $D^m_{\mathbf{Sc},\sigma}$. In Figure 5.1b, which represents our improved sampling algorithm, the distribution from which we are sampling is bimodal having its two centers at $\mathbf{Sc}$ and $-\mathbf{Sc}$. As can be seen from the figure, the distribution D^m_σ fits much "better" (*i.e.* needs to be scaled by a much smaller factor) underneath the bimodal distribution and therefore there is a much smaller rejection area between the two curves. As a side note, whereas in Figure 5.1a, a negligible fraction of the scaled D^m_σ is still above $D^m_{\mathbf{Sc},\sigma}$, in Figure 5.1b, all of the scaled D^m_σ is underneath the bimodal distribution $\frac{1}{2}D^m_{\mathbf{Sc},\sigma} + \frac{1}{2}D^m_{-\mathbf{Sc},\sigma}$.

While the above sampling procedure potentially produces much shorter signatures since the Gaussian "tail-cut" factor τ is never used, it does not give an improved signature scheme by itself because the verification procedure is no longer guaranteed to work. The verification checks

[3] One could be tempted to choose a different target distribution, *e.g.* a discrete Gaussian distribution $D_{\sigma'}$ of standard deviation $\sigma' \neq \sigma$. In that case we need to choose M such that

$$D^m_{\sigma'}(\mathbf{x}) / \left(\frac{1}{2}D^m_{\mathbf{Sc},\sigma}(\mathbf{x}) + \frac{1}{2}D^m_{-\mathbf{Sc},\sigma}(\mathbf{x})\right) \leqslant \frac{\rho_\sigma(\mathbb{Z})}{\rho_{\sigma'}(\mathbb{Z})} \exp\left(\frac{\|\mathbf{Sc}\|^2}{2\sigma^2} + \frac{\|\mathbf{z}\|^2(\sigma'^2 - \sigma^2)}{2(\sigma\sigma')^2}\right) \leqslant M\,,$$

and since $\frac{\rho_\sigma(\mathbb{Z})}{\rho_{\sigma'}(\mathbb{Z})} \approx \frac{\sigma}{\sigma'}$ this yield a larger M either when $\|\mathbf{z}\|^2 < 2\log_e(\sigma'/\sigma)(\sigma\sigma')^2/(\sigma'^2 - \sigma^2)$ when $\sigma' > \sigma$ or when $\|\mathbf{z}\|^2 > 2\log_e(\sigma/\sigma')(\sigma\sigma')^2/(\sigma^2 - \sigma'^2)$ when $\sigma > \sigma'$. It therefore seems that when the target is a discrete Gaussian distribution, choosing $\sigma = \sigma'$ is optimal; however it might be that a differently shaped target distribution yields a smaller rejection rate.

that $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc} \bmod q, \mu)$ and so will verify correctly if and only if $\mathbf{Ay} = \mathbf{Az} - \mathbf{Tc} = \mathbf{A}((-1)^b\mathbf{Sc} + \mathbf{y}) - \mathbf{Tc} = \mathbf{Ay} + (-1)^b\mathbf{Tc} - \mathbf{Tc}$, which will only happen if $(-1)^b\mathbf{Tc} = \mathbf{Tc} \bmod q$ for $b \in \{0, 1\}$. In other words, we will need $\mathbf{Tc} = -\mathbf{Tc} \bmod q$, which will never happen if q is prime unless $\mathbf{T} = \mathbf{0}$.[4] Our solution, therefore, is to work modulo $2q$ and to set $\mathbf{T} = q\mathbf{I}$ where $\mathbf{I}$ is the $n \times n$ identity matrix. In this case $\mathbf{Tc} = -\mathbf{Tc} \bmod 2q$, and so the verification procedure will always work.

Changing the modulus from q to $2q$ and forcing the matrix $\mathbf{T}$ to always be $q\mathbf{I}$ creates several potential problems. In particular, it is no longer clear how to perform key generation, and also the outline for the security proof from [Lyu12] no longer holds. But we show that these problems can be overcome. We will now sketch the key generation and the security proof based on the hardness of the SIS problem in which one is given a uniformly random matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, and is asked to find a short non-zero vector $\mathbf{w}$ such that $\mathbf{Bw} = \mathbf{0} \pmod q$. To generate the public and secret keys, we first pick a uniformly random matrix $\mathbf{A}' \in \mathbb{Z}_q^{n\times(m-n)}$ and a random $(m-n)\times n$ matrix $\mathbf{S}'$ consisting of short coefficients. We then compute $\mathbf{A}'' = \mathbf{A}'\mathbf{S}' \bmod q$ and output $\mathbf{A} = (2\mathbf{A}'|q\mathbf{I} - 2\mathbf{A}'')$ as the public key. The secret key is $\mathbf{S} = (\mathbf{S}'|\mathbf{I})^t$. Note that by construction we have $\mathbf{AS} = q\mathbf{I} \pmod{2q}$ and $\mathbf{S}$ consists of small entries. The dimensions m and n are picked so that the distribution of $(\mathbf{A}'|\mathbf{A}'\mathbf{S}' \bmod q)$ can be shown to be uniformly random in $\mathbb{Z}_q^{n\times m}$ by the Leftover Hash Lemma.

In the security proof, we are given a random matrix $\mathbf{B} = (\mathbf{A}'|\mathbf{A}'') \in \mathbb{Z}_q^{n\times m}$ by the challenger and use the adversary that forges a signature to find a short non-zero vector $\mathbf{w}$ such that $\mathbf{Bw} = \mathbf{0} \pmod q$. We create the public key $\mathbf{A} = (2\mathbf{A}'|q\mathbf{I} - 2\mathbf{A}'')$ and give it to the adversary. Even though we do not know a secret key $\mathbf{S}$ such that $\mathbf{AS} = q\mathbf{I} \pmod{2q}$, we can still create valid signatures for any messages of the adversary's choosing by picking the $(\mathbf{z}, \mathbf{c})$ according to the correct distributions and then programming the random oracle as is done in [Lyu12]. When the adversary forges, we use the forking lemma to create two equations $\mathbf{Az} = q\mathbf{c} \pmod{2q}$ and $\mathbf{Az}' = q\mathbf{c}' \pmod{2q}$. Combining them together, we obtain $\mathbf{A}(\mathbf{z} - \mathbf{z}') = q(\mathbf{c} - \mathbf{c}') \pmod{2q}$. Under some very simple requirements for $\mathbf{z}, \mathbf{z}', \mathbf{c}$, and $\mathbf{c}'$, the previous equation implies that $\mathbf{A}(\mathbf{z} - \mathbf{z}') = \mathbf{0} \pmod q$ and $\mathbf{z} \neq \mathbf{z}'$. This then implies that $2\mathbf{B}(\mathbf{z} - \mathbf{z}') = \mathbf{0} \pmod q$ and since 2 is invertible modulo q, we have found a $\mathbf{w} = (\mathbf{z} - \mathbf{z}')$ such that $\mathbf{Bw} = \mathbf{0} \pmod q$.

The above scheme construction and proof work for SIS and equally well for Ring-SIS, when instantiated with polynomials. As in [Lyu12], we can also construct much more efficient schemes based on LWE and Ring-LWE by creating the matrix $\mathbf{A}'' = \mathbf{A}'\mathbf{S}'$ such that $(\mathbf{A}'|\mathbf{A}'')$ is not uniformly random, but only computationally. For optimal efficiency, though, we can create the key in yet a different manner related to the way NTRU keys are generated [HPS98, HHGP+03]. The formal construction and the implementation are described in Chapter 6.

5.2 Preliminaries

Notation. For any integer q, we identify the ring $\mathbb{Z}_q$ with the interval $[-q/2, q/2) \cap \mathbb{Z}$. We define $\mathbb{B} = \{0, 1\}$ the set of binary integers and $\mathbb{B}_w^n$ the set of binary vectors of length n and Hamming weight w (*i.e.* vectors with exactly w out of n non-zero entries). Vectors, considered as column vectors, will be written in bold lower case letters. Matrices will be written in bold upper case letters. For a positive integer n, we write $\mathbf{I}_n$ to be the identity matrix of dimension n.

We recall that the ℓ_p-norm of a vector $\mathbf{v}$ is defined as $\|\mathbf{v}\|_p = (\sum_i |v_i|^p)^{1/p}$ for $p > 0$, and its ℓ_∞-norm as $\|\mathbf{v}\|_\infty = \max_i |v_i|$. By default, we use $\|\cdot\|$ for the ℓ_2-norm.

Hardness Assumptions. All the constructions in this paper are based on the hardness of the *generalized SIS* (Short Integer Solution) problem, for a ring $R = \mathbb{Z}$ or $R = \mathbb{Z}[x]/(f(x))$ where f is a monic polynomial, which we define below.

Definition 5.1 (R-SIS$^{\mathcal{K}}_{q,n,m,\beta}$ problem)**.** Let $\mathcal{K}$ be some distribution over $R_q^{n\times m}$, where R_q is the quotient ring R/qR. Given a random $\mathbf{A} \in R_q^{n\times m}$ drawn according to the distribution $\mathcal{K}$, find a non-zero $\mathbf{v} \in R_q^m$ such that $\mathbf{Av} = \mathbf{0}$ and $\|\mathbf{v}\|_2 \leqslant \beta$.

[4] One may think that a possible solution could be to output the bit b as part of the signature, but this is not secure. Depending on the sign of $\langle \mathbf{z}, \mathbf{Sc} \rangle$, one of the two values of b is more likely to be output than the other. Therefore outputting the bit b leaks information about $\mathbf{S}$.

If we let $R = \mathbb{Z}$ (resp. $R = \mathbb{Z}[x]/(f(x))$) and $\mathcal{K}$ be the uniform distribution, then the resulting problem is the classical SIS (resp. Ring-SIS) problem – *cf.* Section 3.2.2.

General Forking Lemma. An important tool to prove the security of signature schemes in the random oracle model is the forking lemma, first introduced by Pointcheval and Stern [PS96]. The basic principle is to run an adversary A having non-negligible probability of forging a signature with a random tape r and a specific (challengerially chosen) instance of the random oracle. If successful the forgery corresponds to a specific hash query at index i. The forking lemma states that with non-negligible probability, when we replay the adversary with the same random tape r and answering the first $i-1$ queries to the random oracle with the same values as before but answering the subsequent queries with freshly chosen random values, the forger outputs a new forgery that corresponds again to the i-th hash query. We state below the general formulation of Bellare and Neven [BN06].

Lemma 5.2 (General Forking Lemma)**.** *Fix an integer $t \geqslant 1$ and a set B of size $|B| \geqslant 2$. Let A be a probabilistic algorithm that on inputs $(x, c_1, \ldots, c_t; r)$ outputs a pair $(i \in \{0, \ldots, t\}, y)$ where r is the random tape of A, IG be a probability distribution from which x is drawn and the c_i's are uniformly drawn from B.*

The forking algorithm F_A associated to A is the randomized algorithm that takes as input x and proceeds as follows:

1. *Pick a random tape r for A*
2. *Pick $c_1, \ldots, c_t$ uniformly from B*
3. *Run A on input $(x, c_1, \ldots, c_t; r)$ to produce (i, y)*
4. *If $i = 0$, return $(0, 0, 0)$*
5. *Pick $c'_i, \ldots, c'_t$ uniformly from B*
6. *Run A on input $(x, c_1, \ldots, c_{i-1}, c'_i, \ldots, c'_t; r)$ to produce (i', y')*
7. *If $i = i'$ and $y \neq y'$, return $(1, y, y')$, else return $(0, 0, 0)$*

Let δ be the probability that A outputs a tuple with $i \geqslant 1$ and ε the probability that F_A outputs a triple starting with 1 given an input x randomly generated according to IG. Then

$$\varepsilon \geqslant \delta \cdot \left(\frac{\delta}{t} - \frac{1}{|B|} \right).$$

In our context of digital based signatures, think of x as a public key, $c_1, \ldots, c_t$ as responses to queries to a random oracle or signing oracle, A as an adversary having a non-negligible probability δ to forge a signature y.

5.3 BLISS: A Lattice Signature Scheme using Bimodal Gaussians

In this section, we present our new signature scheme along with the proof of correctness. The security of the signature scheme is based on the hardness of the $R\text{-SIS}^{\mathcal{K}}_{q,n,m,\beta}$ problem. A specific implementation that uses numerous enhancements is presented in Chapter 6. For simplicity, we present our algorithm for $R = \mathbb{Z}$, but it works in exactly the same way for rings $R = \mathbb{Z}[x]/(x^n + 1)$ (used in Chapter 6).

5.3.1 New Signature and Verification Algorithms

Key pairs. The secret key is a (short) matrix $\mathbf{S} \in \mathbb{Z}_{2q}^{m \times n}$ and the public key is given by the matrix $\mathbf{A} \in \mathbb{Z}_{2q}^{n \times m}$ such that $\mathbf{AS} = q\mathbf{I}_n \pmod{2q}$. A crucial property, for our new rejection sampling algorithm, satisfied by the key pair, is that $\mathbf{AS} = \mathbf{A}(-\mathbf{S}) = q\mathbf{I}_n \pmod{2q}$. Obtaining such a key

Algorithm 5.1 Signature Algorithm.

function Sign(Message digest μ, public key $\mathbf{A} \in \mathbb{Z}_{2q}^{n \times m}$, secret key $\mathbf{S} \in \mathbb{Z}_{2q}^{m \times n}$, std. dev. $\sigma \in \mathbb{R}$)
  $\mathbf{y} \leftarrow D_\sigma^m$
  $\mathbf{c} \leftarrow H(\mathbf{A}\mathbf{y} \bmod 2q, \mu)$ ▷ Compute the challenge
  $b \leftarrow \{0, 1\}$
  $\mathbf{z} \leftarrow \mathbf{y} + (-1)^b \mathbf{S}\mathbf{c}$
  $b \leftarrow \mathcal{B}_{1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{S}\mathbf{c}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{S}\mathbf{c} \rangle}{\sigma^2}\right)\right)}$
  if $b = 1$ **then** ▷ output with probability $\frac{1}{M \exp\left(-\frac{\|\mathbf{S}\mathbf{c}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{S}\mathbf{c} \rangle}{\sigma^2}\right)}$
    return $(\mathbf{z}, \mathbf{c})$
  else
    restart
  end if
end function

pair is easy and can be done efficiently. Let us present a key-generation procedure which results in a scheme whose security is based on the classic $\mathrm{SIS}_{q,n,m,\beta}$ problem.[5]

Define $m' = m - n$. Choose a uniform matrix $\mathbf{A}' \in \mathbb{Z}_q^{n \times m'}$ and a random small $\mathbf{S}' \in \mathbb{Z}_q^{m' \times n}$ with coefficients in $(-2^\alpha, 2^\alpha)$. Define $\mathbf{A}'' = \mathbf{A}'\mathbf{S}' \bmod q$. By Lemma 3.4, the statistical distance between the distribution of $\mathbf{A}''$ and the uniform distribution over $\mathbb{Z}_q^{n \times n}$ is at most $n \cdot 1/2\sqrt{q^n / 2^{(\alpha+1) \cdot m'}}$. Thus, for this statistical distance to be negligible in the security parameter λ, we need

$$m \geqslant n + \frac{2(\lambda - 1 + \log_2(n)) + n \log_2(q)}{\alpha + 1} \,. \tag{5.2}$$

Set the secret key as $\mathbf{S} = \begin{pmatrix} \mathbf{S}' \\ \mathbf{I}_n \end{pmatrix} \in \mathbb{Z}_{2q}^{m \times n}$. It remains to set the public key as $\mathbf{A} = (2\mathbf{A}' | q\mathbf{I}_n - 2\mathbf{A}'') \in \mathbb{Z}_{2q}^{n \times m}$. Then one easily checks that $\mathbf{A}\mathbf{S} = q\mathbf{I}_n$. Also, we have that $\mathbf{A} \bmod q$ is *uniform* modulo q. Note that this construction is easily adaptable to the ring settings.

Random Oracle Domain. We model the hash function H as a random oracle that has uniform output in $\mathbb{B}_\kappa^n$, the set of binary vectors of length n and Hamming weight κ. An efficient construction of such a random oracle can be found in Chapter 6.

The Signature Algorithm (Algorithm 5.1). The signer, who is given a message digest μ, first samples a vector $\mathbf{y}$ from the m-dimensional discrete Gaussian distribution D_σ^m and then computes $\mathbf{c} \leftarrow H(\mathbf{A}\mathbf{y} \bmod 2q, \mu)$. Then she samples a bit b in $\{0, 1\}$ and computes the potential output $\mathbf{z} \leftarrow \mathbf{y} + (-1)^b \mathbf{S}\mathbf{c}$. Note that $\mathbf{z}$ is distributed according to the bimodal discrete Gaussian distribution $\frac{1}{2} D_{\mathbf{S}\mathbf{c},\sigma}^m + \frac{1}{2} D_{-\mathbf{S}\mathbf{c},\sigma}^m$. At this point we perform rejection sampling and output the signature $(\mathbf{z}, \mathbf{c})$ with probability

$$1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{S}\mathbf{c}\|^2}{2\sigma^2} \right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{S}\mathbf{c} \rangle}{\sigma^2} \right) \right) \,,$$

where M is some fixed positive real that is set large enough to ensure that the preceding probability is always at most 1. We explain how to set M in accordance with the standard deviation σ in the next subsection. If the signing algorithm did not output the signature, then it is restarted and repeated until something is outputted. The expected number of iterations of the signing algorithm is M.

The Verification Algorithm (Algorithm 5.2). The verification algorithm will accept $(\mathbf{z}, \mathbf{c})$ as the signature for μ if the following three conditions hold:

1. $\|\mathbf{z}\| \quad \leqslant B_2$

[5] In Chapter 6, we present an "NTRU-like" variant of the key generation which yields a more efficient instantiation of the signature scheme.

Algorithm 5.2 Verification Algorithm.

1: **function** VERIFY(Message digest μ, public key $\mathbf{A} \in \mathbb{Z}_{2q}^n$, signature $(\mathbf{z}, \mathbf{c})$)
2: **if** $\|\mathbf{z}\| > B_2$ or $\|\mathbf{z}\|_\infty \geqslant q/4$ **then**
3: **return** Reject
4: **end if**
5: **if** $\mathbf{c} = H(\mathbf{Az} + q\mathbf{c} \bmod 2q, \mu)$ **then**
6: **return** Reject
7: **else**
8: **return** Accept
9: **end if**
10: **end function**

2. $\|\mathbf{z}\|_\infty < q/4$

3. $\mathbf{c} = H(\mathbf{Az} + q\mathbf{c} \bmod 2q, \mu)$

The signer outputs signatures of the form $(\mathbf{z}, \mathbf{c})$ where $\mathbf{z}$ is distributed according to D_σ^m, thus the acceptance bound B_2 should be set a little bit higher than $\sqrt{m}\sigma$, which is the expected value around which the output of D_σ^m is tightly concentrated; denoting $B_2 = \eta\sqrt{m}\sigma$, one can set η so that $\|\mathbf{z}\| \leqslant B_2$ is verified with probability $1 - 2^{-\lambda}$ [Lyu12, Lemma 4.4] for the security parameter λ (in practice, $\eta \in [1.1, 1.4]$). For technical reasons in the security proof, we also need that $\|\mathbf{z}\|_\infty < q/4$, but this condition is usually verified whenever the first one is and does not restrict the manner in which we choose the parameters for the scheme. Condition 3 will also hold for valid signatures because

$$\mathbf{Az} + q\mathbf{c} = \mathbf{A}(\mathbf{y} + (-1)^b\mathbf{Sc}) + q\mathbf{c} = \mathbf{Ay} + \big((-1)^b\mathbf{AS}\big)\mathbf{c} + q\mathbf{c} = \mathbf{Ay} + (q\mathbf{I}_n)\mathbf{c} + q\mathbf{c} = \mathbf{Ay} \bmod 2q\,.$$

5.3.2 Rejection Sampling: Correctness and Efficiency

We now explain how to pick the standard deviation σ and positive real M so that the signing algorithm in the preceding section produces vectors $\mathbf{z}$ according to the distribution D_σ^m. Because $\mathbf{y}$ is distributed according to D_σ^m, it is easy to see that in Step 5 of the signing algorithm, $\mathbf{z}$ is distributed according to $g_{\mathbf{Sc}} = \frac{1}{2}D^m_{\mathbf{Sc},\sigma} + \frac{1}{2}D^m_{-\mathbf{Sc},\sigma}$ for fixed $\mathbf{Sc}$ and over the space of all $(b, \mathbf{y})$. Thus for any $\mathbf{z}^* \in \mathbb{R}^m$, we have

$$\begin{aligned}
\Pr[\mathbf{z} = \mathbf{z}^*] &= \frac{1}{2}D^m_{\mathbf{Sc},\sigma}(\mathbf{z}^*) + \frac{1}{2}D^m_{-\mathbf{Sc},\sigma}(\mathbf{z}^*) \\
&= \frac{1}{2\rho_\sigma(\mathbb{Z}^m)}\exp\left(-\frac{\|\mathbf{z}^* - \mathbf{Sc}\|^2}{2\sigma^2}\right) + \frac{1}{2\rho_\sigma(\mathbb{Z}^m)}\exp\left(-\frac{\|\mathbf{z}^* + \mathbf{Sc}\|^2}{2\sigma^2}\right) \\
&= \frac{1}{2\rho_\sigma(\mathbb{Z}^m)}\exp\left(-\frac{\|\mathbf{z}^*\|^2}{2\sigma^2}\right)\exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right)\left(e^{-\frac{\langle \mathbf{z}^*, \mathbf{Sc}\rangle}{\sigma^2}} + e^{\frac{\langle \mathbf{z}^*, \mathbf{Sc}\rangle}{\sigma^2}}\right) \\
&= \frac{1}{\rho_\sigma(\mathbb{Z}^m)}\exp\left(-\frac{\|\mathbf{z}^*\|^2}{2\sigma^2}\right)\exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right)\cosh\left(\frac{\langle \mathbf{z}^*, \mathbf{Sc}\rangle}{\sigma^2}\right)\,.
\end{aligned}$$

The desired output distribution is the centered discrete Gaussian distribution, whose probability distribution is $f(\mathbf{z}^*) = \rho_\sigma(\mathbf{z}^*)/\rho_\sigma(\mathbb{Z}^m)$. Thus, by Lemma 3.5, one should accept the sample $\mathbf{z}^*$ with probability:

$$p_{\mathbf{z}*} = \frac{f(\mathbf{z}^*)}{Mg_{\mathbf{Sc}}(\mathbf{z}^*)} = 1\Big/\left(M\exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right)\cosh\left(\frac{\langle \mathbf{z}^*, \mathbf{Sc}\rangle}{\sigma^2}\right)\right)\,,$$

where M is chosen large enough so that $p_{\mathbf{z}*} \leqslant 1$. Note that $\cosh(x) \geqslant 1$ for any x, so it suffices that

$$M = e^{\frac{1}{2\alpha^2}} \tag{5.3}$$

where α is such that $\sigma \geqslant \alpha \cdot \max_{\mathbf{S},\mathbf{c}}\|\mathbf{Sc}\|$.

Comparison with [Lyu12]. In the original scheme, denoting $\sigma = \alpha \cdot \max_{\mathbf{S},\mathbf{c}} \|\mathbf{Sc}\|$, the repetition rate M was given by:

$$M = \exp\left(12/\alpha + 1/(2\alpha^2)\right),$$

and $\|\mathbf{Sc}\|$ was bounded by $\kappa\|\mathbf{S}\|$. In practice, keeping the repetition rate at the same level (*i.e.* between $e^2 \approx 7.2$ and $e^{1/2} \approx 1.6$), our technique divides the parameter α (and thus σ) by a factor 12 to 24, yielding a great impact on the parameters. Note that this improvement is *not constant* with the security parameter. Indeed, if one wishes a statistical distance of $2^{-\lambda}$, the rejection in [Lyu12] requires $M = \exp(\tau/\alpha + 1/2\alpha^2)$ with $\tau = \mathcal{O}(\sqrt{\lambda})$, while there is no such dependence on our scheme.

Bound on $\|\mathbf{Sc}\|$. Note that if we fix the repetition rate M, then the standard deviation of the signature $\mathbf{z}$, and therefore also its size, only depend on the maximum possible norm of the vector $\mathbf{Sc}$. For this reason, it is important to obtain a bound as tight as possible on this product. Several upper bounds on $\|\mathbf{Sc}\|$ can be used such as $\|\mathbf{Sc}\| \leqslant \|\mathbf{c}\|_1 \cdot \|\mathbf{S}\| = \kappa\|\mathbf{S}\|$ (as in [Lyu12]) or $\|\mathbf{Sc}\| \leqslant s_1(\mathbf{S}) \cdot \|\mathbf{c}\| = s_1(\mathbf{S}) \cdot \sqrt{\kappa}$ where $s_1(\mathbf{S})$ is the singular norm of $\mathbf{S}$. In Chapter 6, we introduce a new measure of $\mathbf{S}$ adapted to the form of $\mathbf{c}$ which helps us achieve a tighter bound than with previous methods.

5.3.3 Security Proof

Any existential forger against our signature scheme can solve the $\mathrm{SIS}^{\mathcal{K}}_{q,n,m,\beta}$ problem for $\beta = 2B_2$ where $\mathcal{K}$ is the distribution induced by the public-key generation algorithm.

Theorem 5.3. *Suppose there is a polynomial-time algorithm $\mathcal{F}$ which makes at most s queries to the signing oracle and h queries to the random oracle H, and succeeds in forging with non negligible probability δ. Then there exists a polynomial-time algorithm which can solve the $SIS^{\mathcal{K}}_{q,n,m,\beta}$ problem for $\beta = 2B_2$ with probability $\approx \frac{\delta^2}{2(h+s)}$. Moreover the signing algorithm produces a signature with probability $\approx 1/M$ and the verifying algorithm accepts a signature produced by an honest signer with probability at least $1 - 2^m$.*

The proof of the theorem follows from standard arguments, and is simpler and tighter than the proof of [Lyu12]. In a nutshell, the fact that the distribution of the signatures in the scheme does not depend on the secret key means that the simulator can "sign" arbitrary messages without having the secret key by programming the random oracle. Then when the adversary produces a forgery, the simulator can extract a solution to the SIS problem. It is proved in a sequence of two lemmas. In Lemma 5.4, we show that our signing algorithm (not restarted) can be replaced with $\textsc{Hybrid}_2$ (Algorithm 5.4), and the statistical distance between the two outputs will be at most $\epsilon = s(s+h)2^{-n+1}$. Since $\textsc{Hybrid}_2$ produces an output with probability exactly $1/M$, the signing algorithm (not restarted) produces an output with probability at least $(1-\epsilon)/M$. Then in Lemma 5.5 we show that if a forger can produce a forgery with probability δ when the signing algorithm is replaced with $\textsc{Hybrid}_2$ (restarted if it does not output anything), then we can use it to recover a vector $\mathbf{v} \neq \mathbf{0}$ such that $\|\mathbf{v}\| \leqslant \beta = 2B_2$ and $\mathbf{Av} = \mathbf{0} \bmod q$ with probability at least $\delta^2/(2(s+h))$.

Algorithm 5.3 HYBRID$_1$

function HYBRID$_1$(Message digest μ, public key $\mathbf{A} \in \mathbb{Z}_{2q}^{n \times m}$, secret key $\mathbf{S} \in \mathbb{Z}_{2q}^{m \times n}$, std. dev. $\sigma \in \mathbb{R}$)
  $\mathbf{y} \leftarrow D_\sigma^m$
  $\mathbf{c} \leftarrow \mathbb{B}_\kappa^n$
  $b \leftarrow \{0,1\}$
  $\mathbf{z} \leftarrow \mathbf{y} + (-1)^b \mathbf{S}\mathbf{c}$
  $b \leftarrow \mathcal{B}_{1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{S}\mathbf{c}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{S}\mathbf{c} \rangle}{\sigma^2}\right)\right)}$
  if $b = 1$ **then** ▷ output with probability $\frac{1}{M \exp\left(-\frac{\|\mathbf{S}\mathbf{c}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{S}\mathbf{c} \rangle}{\sigma^2}\right)}$
    program $H(\mathbf{A}\mathbf{z} + q\mathbf{c}, \mu) = \mathbf{c}$
    return $(\mathbf{z}, \mathbf{c})$
  end if
end function

Algorithm 5.4 HYBRID$_2$

function HYBRID$_2$(Message digest μ, public key $\mathbf{A} \in \mathbb{Z}_{2q}^{n \times m}$, std. dev. $\sigma \in \mathbb{R}$)
  $\mathbf{c} \leftarrow \mathbb{B}_\kappa^n$
  $\mathbf{z} \leftarrow D_\sigma^m$
  $b \leftarrow \mathcal{B}_{1/M}$
  if $b = 1$ **then** ▷ output with probability $1/M$
    program $H(\mathbf{A}\mathbf{z} + q\mathbf{c}, \mu) = \mathbf{c}$
    return $(\mathbf{z}, \mathbf{c})$
  end if
end function

Lemma 5.4. *Let $\mathcal{D}$ be a distinguisher which can query the random oracle H and either the actual signing algorithm (Algorithm 5.1) without the restarting step, or* Hybrid$_2$ *(Algorithm 5.4). If she makes h queries to H and s queries to the signing algorithm that she has access to, then for all but a $1 - e^{-\Omega(n)}$ fraction of all possible matrices $\mathbf{A}$, her advantage in distinguishing the actual signing algorithm (not restarted) from the one in* Hybrid$_2$ *is at most $s(s+h)2^{-n+1}$.*

Proof. First, we show that the distinguisher $\mathcal{D}$ has advantage at most $s(s+h)2^{-n+1}$ in distinguishing the real signature scheme (not restarted) from an output of HYBRID$_1$ (Algorithm 5.3). The only difference between these algorithms is that, in HYBRID$_1$, the output of the random oracle is chosen at random from $\mathbb{B}_\kappa^n$ and then programmed as the answer to $H(\mathbf{A}\mathbf{z} + q\mathbf{c}, \mu) = H(\mathbf{A}\mathbf{y}, \mu)$ without checking whether the value of $(\mathbf{A}\mathbf{y}, \mu)$ was already set. Now, each time HYBRID$_1$ is called, the probability of generating a $\mathbf{y}$ such that $\mathbf{A}\mathbf{y}$ is equal to one of the previous values that was queried is at most 2^{-n+1}. Indeed, let us note that at most $s + h$ values of $(\mathbf{A}\mathbf{y}, \mu)$ will ever be set. With probability at least $1 - e^{\Omega(n)}$, the matrix $\mathbf{A}$ can be written in Hermite Normal Form as $\mathbf{A} = [\bar{\mathbf{A}} \| \mathbf{I}]$. Finally, for any $\mathbf{t} \in \mathbb{Z}_{2q}^n$, since $\sigma \geqslant 3/\sqrt{2\pi}$, we have

$$\begin{aligned}\Pr[\mathbf{A}\mathbf{y} = \mathbf{t};\ \mathbf{y} \leftarrow D_\sigma^m] &= \Pr[\mathbf{y_1} = (\mathbf{t} - \bar{\mathbf{A}}\mathbf{y}_0);\ \mathbf{y} = (\mathbf{y}_0, \mathbf{y}_1) \leftarrow D_\sigma^m] \\ &\leqslant \max_{\mathbf{t}' \in \mathbb{Z}_{2q}^n} \Pr[\mathbf{y_1} = \mathbf{t}';\ \mathbf{y_1} \leftarrow D_\sigma^n] \leqslant 2^{-n}\ .\end{aligned}$$

Thus if HYBRID$_1$ is accessed s times, and the probability of getting a collision each time is at most $(s+h)2^{-n+1}$, the probability that a collision occurs after s queries is at most $s(s+h)2^{-n+1}$.

We next emphasize that the outputs of HYBRID$_1$ and HYBRID$_2$ exactly follows the same distribution. This is a direct consequence of Lemma 3.5: HYBRID$_1$ exactly plays the role of algorithm $\mathcal{A}$ and HYBRID$_2$ corresponds to $\mathcal{F}$, where $M = \exp(1/(2\alpha^2))$,

$$f(\mathbf{z}) = \exp\big(-\|\mathbf{z}\|^2/(2\sigma^2)\big)/\rho_\sigma(\mathbb{Z})$$

and

$$g_{\mathbf{c}}(\mathbf{z}) = \exp\big(-\|\mathbf{z}\|^2/(2\sigma^2)\big)\exp\big(-\|\mathbf{Sc}\|^2/(2\sigma^2)\big)\cosh\big(\langle\mathbf{z},\mathbf{Sc}\rangle/\sigma^2\big)/\rho_\sigma(\mathbb{Z})\ .$$

By Lemma 3.5 the outputs of HYBRID_1 and HYBRID_2 follow the same distribution (since we have $M \cdot g_{\mathbf{c}} \geqslant f$ for all $\mathbf{v}$). □

Lemma 5.5. *Suppose there exists a polynomial-time algorithm $\mathcal{F}$ which makes at most s queries to the signer in* Hybrid_2 *(restarted if necessary), h queries to the random oracle H, and succeeds in forging with probability δ. Then there exists an algorithm with the same time-complexity as $\mathcal{F}$ which, for a given $\mathbf{B} \leftarrow \mathcal{K}$, finds with probability at least $\approx \delta^2/(2(s+h))$ a non-zero $\mathbf{v} \in \mathbb{Z}^m$ such that $\|\mathbf{v}\| \leqslant 2B_2$ and $\mathbf{Bv} = 0$.*

Proof. Let $\mathbf{B} = (\mathbf{A}'|-\mathbf{A}'') \leftarrow \mathcal{K}$ be the matrix for the generalized SIS instance we want to solve where $\mathbf{A}' \in \mathbb{Z}_q^{n\times(m-n)}$ and $\mathbf{A}'' \in \mathbb{Z}_q^{n\times n}$. We define the public key $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ such that $\mathbf{A} = (2\mathbf{A}'|q\mathbf{I}_n - 2\mathbf{A}'')$; note that this modification is such that $\mathbf{A} \bmod q = 2\mathbf{B}$ for our key generation procedures. Therefore finding a vector $\mathbf{v}$ such that $\mathbf{Av} = \mathbf{0} \bmod q$ yields $\mathbf{Bv} = \mathbf{0} \bmod q$ because 2 is invertible modulo q.[6] Denote by $t = s + h$ the bound on the number of times the random oracle H is called or programmed during $\mathcal{F}$'s attack.

First, we pick random coins ϕ and ψ respectively for the forger and the signer. We also pick the values that will correspond to the responses of the random oracle $\mathbf{c}_1, \ldots, \mathbf{c}_t \leftarrow \mathbb{B}_\kappa^n$. We now consider a subroutine $\mathcal{A}$ taking as input $(\mathbf{A}, \phi, \psi, \mathbf{c}_1, \ldots, \mathbf{c}_t)$.

The first step of the subroutine is to initialize $\mathcal{F}$ by giving it the public-key $\mathbf{A}$ and the random coins ϕ. Then, it proceeds to run $\mathcal{F}$. Whenever $\mathcal{F}$ wants some message signed, $\mathcal{A}$ runs the signing algorithm of HYBRID_2 using the signer random coins ψ to produce a signature. During signing or when $\mathcal{F}$ will make queries to the random oracle, the random oracle H will have to be programmed, and the response of H will be the first $\mathbf{c}_i$ in the list $(\mathbf{c}_1, \ldots, \mathbf{c}_t)$ that has not been used yet. (Of course, $\mathcal{A}$ keeps a table of all queries to H, so in case the same query is made twice, the previously answered $\mathbf{c}_i$ will be replied.) When $\mathcal{F}$ finishes running and outputs a forgery (with probability δ), our subroutine $\mathcal{A}$ simply outputs $\mathcal{F}$'s output $\{(\mathbf{z}, \mathbf{c}), \mu\}$.

Recall that the output of $\mathcal{A}$ verifies $\|\mathbf{z}\|_\infty < q/4$ and $\|\mathbf{z}\| \leqslant B_2$ and $\mathbf{c} = H(\mathbf{Az} + q\mathbf{c}, \mu)$. Note that if the random oracle H was not queried or programmed on some input $\mathbf{w} = \mathbf{Az} + q\mathbf{c}$, then $\mathcal{F}$ has only a $1/|\mathbb{B}_\kappa^n|$ chance of producing a $\mathbf{c}$ such that $\mathbf{c} = H(\mathbf{w}, \mu)$. Thus with probability $1 - 1/|\mathbb{B}_\kappa^n|$, $\mathbf{c}$ must be one of the $\mathbf{c}_i$'s, and so the probability that $\mathcal{F}$ succeeds in a forgery and that $\mathbf{c} = \mathbf{c}_j$ for some j is $\delta - \delta/|\mathbb{B}_\kappa^n|$.

Type 1 Forgery. Suppose that $\mathbf{c}_j$ was a response to a signing query made by $\mathcal{F}$ on $(\mathbf{w}', \mu') = (\mathbf{Az}' + q\mathbf{c}_j, \mu')$. Then we would have

$$H(\mathbf{Az} + q\mathbf{c}_j, \mu) = H(\mathbf{Az}' + q\mathbf{c}_j, \mu').$$

If $\mu \neq \mu'$ or $\mathbf{Az} + q\mathbf{c}_j \neq \mathbf{Az}' + q\mathbf{c}_j$, it means that $\mathcal{F}$ found a pre-image of $\mathbf{c}_j$. Therefore with overwhelming probability, we have $\mu = \mu'$ and $\mathbf{Az} + q\mathbf{c}_j = \mathbf{Az}' + q\mathbf{c}_j$. This yields $\mathbf{A}(\mathbf{z} - \mathbf{z}') = \mathbf{0} \bmod 2q$. We know that $\mathbf{z} \neq \mathbf{z}'$ (otherwise the signatures would be the same). Moreover, since $\|\mathbf{z}\|_\infty, \|\mathbf{z}'\|_\infty < q/4$, we have $\mathbf{z} - \mathbf{z}' \neq 0 \bmod q$. Finally, the condition on the ℓ_2-norm of $\mathbf{z}$ and $\mathbf{z}'$ gives $\|\mathbf{z} - \mathbf{z}'\| \leqslant 2B_2$.

Type 2 Forgery. Assume now that $\mathbf{c}_j$ was a response to a random oracle query made by $\mathcal{F}$. In this case we record this signature $(\mathbf{z}, \mathbf{c}_j)$ on the message μ, and we generate fresh random elements $\mathbf{c'}_j, \ldots, \mathbf{c'}_t \leftarrow \mathbb{B}_\kappa^n$. By the General Forking Lemma of Bellare and Neven [BN06] (Lemma 5.2), we obtain that the probability that $\mathbf{c'}_j \neq \mathbf{c}_j$ and the forger uses the random oracle response $\mathbf{c'}_j$ (and the query associated to it) in the forgery is at least

$$\Big(\delta - \frac{\delta}{|\mathbb{B}_\kappa^n|}\Big) \cdot \Big(\frac{\delta - \delta/|\mathbb{B}_\kappa^n|}{t} - \frac{1}{|\mathbb{B}_\kappa^n|}\Big)\ .$$

[6] For the "NTRU-like" key generation used to instantiate our scheme in Chapter 6, we get from the NTRU SIS assumption a matrix $\mathbf{B} = (\mathbf{B}_1|-\mathbf{B}_2) = (\mathbf{a}|-1)$ where $\mathbf{a} = (2\mathbf{g}+1)/\mathbf{f}$ and we define $\mathbf{A} = (2\mathbf{B}_1|q\mathbf{1} - 2\mathbf{B}_2) = (2\mathbf{a}|q-2)$ that is, a public key with the same distribution as in Chapter 6. Moreover we get $\mathbf{A} \bmod q = 2\mathbf{B} = (2\mathbf{a}|-2)$.

Table 5.1 – Naive Signature Schemes Parameters. The parameters with parameters set III are based on the hardness of the $\mathrm{SIS}_{q,n,m,\beta}$ problem and parameters set IV are based on the hardness of the $\mathrm{SIS}_{q,n,m,\beta}$ search problem. The root Hermite factor for all the instantiations is $\delta = 1.007$.

	BLISS	[Lyu12, **Set-III**]	BLISS	[Lyu12, **Set-IV**]
n	512		512	
d	31		1	
η	1.2		1.3	
κ	17	14	17	14
q	$2^{32.5}$	2^{33}	2^{15}	2^{18}
$M = \exp(1/(2\alpha^2))$	2.72		7.4	
$m \approx 64 + n \cdot \log q / \log(2d+1)$	–	3253	–	–
m (Equation (5.2))	3343	–	–	–
$m = 2n$	–	–	1024	
σ	**21540**	**300926**	**272**	**2688**
β	$2^{21.5}$	$2^{25.4}$	$2^{14.5}$	$2^{17.8}$
approximate signature size $\approx m \cdot \log(12\sigma)$	**60000**	**73000**	**11953**	**15337**
approximate sk size $\approx m \cdot n \cdot \log(2d+1)$	$2^{23.5}$	$2^{23.5}$	$2^{19.5}$	$2^{19.5}$
approximate pk size $\approx (n \cdot m + n \cdot n) \cdot \log q$	$2^{26.5}$	2^{26}	2^{24}	$2^{24.5}$

Thus, with the above probability, $\mathcal{F}$ outputs a signature $(\mathbf{z}', \mathbf{c}'_\mathbf{j})$ of the message μ and $\mathbf{Az} + q\mathbf{c}_j = \mathbf{Az}' + q\mathbf{c}'_\mathbf{j}$. We finally obtain

$$\mathbf{A}\big(\mathbf{z} - \mathbf{z}'\big) = q\big(\mathbf{c}_j - \mathbf{c'}_j\big) \bmod 2q \ .$$

Since $\mathbf{c}_j - \mathbf{c'}_j \neq \mathbf{0} \bmod 2$, we have $\mathbf{z} - \mathbf{z}' \neq \mathbf{0} \bmod 2q$. Moreover, we have $\|\mathbf{z} - \mathbf{z}'\|_\infty < q/2$: this implies that $\mathbf{v} = \mathbf{z} - \mathbf{z}' \neq \mathbf{0} \bmod q$. Finally, we have

$$\mathbf{Av} = \mathbf{0} \bmod q \qquad \text{and} \qquad \|\mathbf{v}\| \leqslant 2B_2 \ ,$$

that is $\mathbf{v}$ is a solution to a $\mathrm{SIS}^{\mathcal{K}}_{q,n,m,\beta}$ with $\beta = 2B_2$. □

5.4 Conclusion

In this chapter we proposed a modification of the rejection sampling algorithm used in Lyubashevsky's signature scheme of Eurocrypt 2012 [Lyu12]. By sampling from a bimodal Gaussian distribution, we reduced the standard deviation of the resulting signatures by a factor that is asymptotically square root in the security parameter, and obtained a scheme with a tighter security proof.

For the sake of comparison, we provide in Table 5.1 parameters for our modified scheme BLISS, using approximately the same parameters as in [Lyu12] for a target security level of 100 bits. In particular, we kept the rejection rate M, the approximation factor η such that $B_2 = \eta\sigma\sqrt{m}$, the target root Hermite factor $\delta = 1.007$ and the secret key distribution $S \leftarrow \{-d, \ldots, d\}^{m \times n}$. We had to modify the Hamming weight of the outputs of the random oracle from 14 to 17 (because [Lyu12] works with ternary vectors) and we updated the values of σ and q thanks to our improved algorithm. For $m = 2n$, the signature schemes are based on the *low-density* $\mathrm{SIS}_{q,n,m,\beta}$ problem.

As emphasized in Table 5.1, our new rejection sampling technique allowed to reduce the standard deviation σ by one order of magnitude, and the signature size by roughly 20% compared to [Lyu12]. Although theoretically and practically meaningful, the resulting schemes are quite frustrating. First and foremost, the resulting sizes are still too large to become competitive with RSA or ECDSA. Even the ring-based variants of the schemes, dividing the key sizes by a factor $n = 512$, yield a public key of 8kB compared to the 0.5kB of RSA and the 0.02kB of ECDSA. Even the signature is still at least three times larger than RSA signatures (and more than 30 times larger than ECDSA signatures) for a comparable security level. On the practical side, the signing algorithm restarts on average 7.4 times for BLISS with the parameters set IV, which impacts the practical performances.

Additionally, the parameters are chosen according to a root Hermite factor $\delta = 1.007$ for an approximate security level of 100 bits [Lyu12], *i.e.* less than the 128 bits suggested to offer long-term cryptographic protection [ECR12, NIS11].

Consequently, provably secure lattice-based cryptography still appears not to be competitive enough for practical applications. In order to prove this assertion wrong, we instantiate an optimized variant of BLISS in Chapter 6, with a key generation based on NTRU lattices [HPS98, HHGP$^+$03], efficient hashing and signature compression (similarly to [GLP12]). We provide an extensive security analysis using the most up-to-date lattice reduction results [CN11] to propose parameters for security levels of 128, 160 and 192 bits. We produce proof-of-concept implementations of the resulting signature schemes that compete in terms of efficiency to RSA and ECDSA.

Chapter 6

Implementation of BLISS

6.1 Introduction

This chapter instantiates BLISS (Bimodal Lattice Signature Scheme), the lattice-based signature scheme described in Chapter 5, to illustrate the practicality of lattice-based cryptography in terms of efficiency and size compared to classical non-quantum primitives such as RSA or ECDSA. We implement a family of digital signature schemes for security levels of 128, 160 and 192 bits on a 64-bit architecture. Our proof-of-concept implementations compare very favorably to the `openssl` [YH13] implementations of RSA and ECDSA[1] signature schemes – in particular our signature scheme is one order of magnitude faster than RSA to sign (and as fast as ECDSA), and one order of magnitude faster than ECDSA to verify (and faster than RSA). In addition, our scheme has shorter signature and public key sizes than all previously proposed lattice signature schemes.

This chapter was part of the article Lattice Signature and Bimodal Gaussian [DDLL13a], cosigned with L. Ducas, A. Durmus and V. Lyubashevsky and published at Crypto 2013 [CG13a]. The full version of the article is available at [DDLL13b]. The proof-of-concept implementations of BLISS are available under license CeCILL at [DL13].

Background. Few concrete parameters have been suggested for lattice-based cryptosystems. With the notable exception of NTRU [HPS98], public keys, ciphertexts or signatures are often very large (several kilobytes/megabytes). Moreover, as already emphasized in Chapter 4, the cryptosystems often require sampling from a discrete Gaussian distribution over a lattice. Therefore, it is still not clear how efficient the schemes based on modern lattice assumptions will be in practice.

In Chapter 4, we proposed an efficient algorithm to sample according to a discrete Gaussian distribution over the integers on constrained devices. In Chapter 5, we improved the signature scheme of Lyubashevsky [Lyu12], believed to be the most efficient lattice-based signature scheme. However as emphasized in Section 5.4, the latter works were not sufficient enough to provide viable alternatives to RSA or ECDSA. A first step towards a practical (secure) lattice-based signature was described by Güneysu, Lyubashevsky and Pöppelmann [GLP12], and further improved by Güneysu, Oder, Pöppelmann and Schwabe [GOPS13]. Unfortunately, we show in this chapter that the underlying scheme is not strongly unforgeable[2] and offers rather 80 bits of security than 100 bits. Moreover, it relies on weaker security assumption than [Lyu12], and the signature has more than 9000 bits.

Our Results and Techniques. In this chapter, we improve further the BLISS scheme described in Chapter 5. Our improvements yield a very promising candidate for message authentication and digital signatures, and prove that lattice-based cryptography is competitive enough for practical applications. In particular, our proof-of-concept implementations compare very favorably to existing schemes such as RSA and ECDSA in terms of efficiency, and our signature is only slightly more than 5000-bit long. We make our implementations publicly available [DL13] under an open-source license to spur the community help lattice-based cryptography become practical.

[1] ECDSA on a prime field $\mathbb{F}_p$.

[2] Strong unforgeability ensures the adversary cannot even produce a new signature for a previously signed message.

More precisely, we create the public and secret keys in yet a different manner related to the way NTRU keys are generated. The formal construction is described in Section 6.2, and we just give the intuition here. We could create two small polynomials $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}[x]/(x^n + 1)$ and output the public key as $\mathbf{a} = \frac{q-\mathbf{s}_2}{\mathbf{s}_1} \pmod{2q}$. Note that this implies that $\mathbf{a}\mathbf{s}_1 + \mathbf{s}_2 = q \pmod{2q}$, and so we can think of the public key as $\mathbf{A} = (\mathbf{a}, 1)$ and the secret key as $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^t$. Assuming that it is a hard problem to find small vectors $\mathbf{w}$ such that $\mathbf{A}\mathbf{w} = \mathbf{0} \pmod{2q}$, the signature scheme instantiated in the above manner will be secure. To those readers familiar with the key generation in the NTRU encryption scheme, the above key generation should look very familiar, except that the modulus is $2q$ rather than q. Since we are not sure what happens when the modulus is $2q$ when q is prime, we show in Section 6.2 how to instantiate our scheme so that it is based on NTRU over modulus q prime. We then explain how for certain instantiations, this is as hard a problem as Ring-SIS (using the results of Stehlé, Steinfeld [SS11b]) and how for more efficient instantiations, it is a *weaker* assumption than the ones underlying the classic NTRU encryption scheme and the recent construction of fully-homomorphic encryption [LTV12].

Previous cryptanalytic efforts against schemes based on SIS and LWE mostly involved computing the Hermite factor of the underlying average-case instance, as in the work of Gama and Nguyen [GN08], and making sure that its value is below the level required for the desired security guarantees. In this chapter we undertake a more careful cryptanalysis by using the results on BKZ-2.0 of Chen and Nguyen [CN11] in combination with other techniques – namely dual lattice reduction and the combinatorial meet-in-the-middle attack of Howgrave-Graham [HG07]. This extensive cryptanalytic survey allowed us to derive parameters for target security levels.

Finally, for optimal efficiency the security of our scheme relies on the hardness of a type of NTRU problem that has recently (re-)appeared in the literature [LTV12] and which, we believe, could play a major role in the future of lattice-based cryptography. The only cryptanalysis of which we are aware of that studies NTRU lattices deals with instances where the modulus is very close in size to the dimension of the lattice [GN08, HHGPW10]. It is thus unclear as to what roles each of the variables plays when looked at independently. In our work, and also in the previously-mentioned work of [LTV12], the modulus is required to be substantially larger than the dimension. As far as we are aware, no previous cryptanalysis was done for these types of instances. In this chapter, we describe results of experiments similar to [GN08], using BKZ-20 in the case of $2n$-dimensional NTRU lattices. It seems from our experiments that the ratio between the Gaussian heuristic and the actual length of the vector dictates the hardness of finding short vectors in NTRU lattices.

6.2 NTRU-Based Key Generation

Throughout this chapter n is a power of two so that $f(x) = x^n + 1$ is a monic irreducible polynomial (the cyclotomic polynomial of order $2n$). Also q is a prime number such that $q = 1 \bmod 2n$. Finally, we work over the rings $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ and $R_{2q} = \mathbb{Z}_{2q}[x]/(x^n + 1)$.

6.2.1 NTRU Lattices

In the NTRU cryptosystem over the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ [HPS98], the key generation procedure picks two short secret keys $\mathbf{f}, \mathbf{g} \in R_q$ (according to some distribution) with $\mathbf{f}$ invertible and computes the public key as $\mathbf{a} = \mathbf{g}/\mathbf{f}$.[3] When the norm of $\mathbf{f}, \mathbf{g}$ is large enough, it can be shown that $\mathbf{a}$ is actually uniformly random in R_q [SS11b], but even when the secret keys do not have enough entropy, their quotient still appears to be pseudorandom, although no proof of this fact is known [LTV12]. In the NTRU cryptosystem (or its more secure modification of [SS11b] which is based on the Ring-LWE problem), one encrypts a message μ, represented as a polynomial in R_q with $\{0, 1\}$ coefficients, by picking two short vectors $\mathbf{r}, \mathbf{e} \in R_q$ and outputting $\mathbf{z} = 2(\mathbf{a}\mathbf{r} + \mathbf{e}) + \mu$. The security of the scheme relies on the fact that the distribution of $(\mathbf{a}, \mathbf{z})$ is pseudo-random in R_q^2.

One can define an NTRU version of the SIS problem that is at least as hard as breaking the NTRU cryptosystem:

[3] In the original NTRU scheme, the ring was $\mathbb{Z}_q[x]/(x^n - 1)$, but lately researchers have also used $\mathbb{Z}_q[x]/(x^n + 1)$ when n is a power of 2. Indeed, the latter choice seems at least as secure.

Definition 6.1 (NTRU SIS)**.** Set $R = \mathbb{Z}[x]/(x^n + 1)$ and let $\mathcal{K}$ be the distribution that picks small $\mathbf{f}, \mathbf{g}$ and outputs the public key $\mathbf{A} = (\mathbf{a}, 1) \in R_q^{1\times 2}$ for $\mathbf{a} = \mathbf{g}/\mathbf{f}$. The NTRU SIS problem with parameters (q, β) is the $R\text{-SIS}^{\mathcal{K}}_{q,1,2,\beta}$ problem (as defined in Definition 5.1).

In particular, given an NTRU public key $\mathbf{a}$, one has to find two polynomials $\mathbf{v}_1, \mathbf{v}_2 \in R_q$ such that $\|(\mathbf{v}_1, \mathbf{v}_2)^t\| \leqslant \beta$ and $\mathbf{a}\mathbf{v}_1 + \mathbf{v}_2 = 0$ in R_q. Note that $(\mathbf{f}, -\mathbf{g})^t$ is a solution to this problem, but in fact, finding larger solutions can also be useful in breaking the NTRU cryptosystem. In particular, note that for any solution $(\mathbf{v}_1, \mathbf{v}_2)^t$, one can compute $\mathbf{z}\mathbf{v}_1 = 2(-\mathbf{r}\mathbf{v}_2 + \mathbf{e}\mathbf{v}_1) + \mu\mathbf{v}_1$. If β is sufficiently small with respect to $\|(\mathbf{r}, \mathbf{e})^t\|$, then $\mathbf{z} \cdot \mathbf{v}_1 \bmod 2 = \mu\mathbf{v}_1$, and μ can be recovered. Thus, for certain parameters, the NTRU SIS problem is at least as hard as breaking the NTRU cryptosystem. As a side-note, we would like to point out that the NTRU encryption scheme remains hard even after 16 years of cryptanalysis. The weakness in the NTRU signature scheme, which uses the same key generation procedure, is due to the fact that signatures slowly leak the secret key [NR09, MPSW09, DN12b]. This is provably (because information-theoretically) avoided in our scheme.

In Section 6.4, we analyze the hardness of the NTRU SIS problem using combinations of lattice [CN11] and hybrid attacks [HG07].

6.2.2 Key-Generation

Given densities δ_1 and δ_2, we generate random polynomials $\mathbf{f}$ and $\mathbf{g}$ with $d_1 = \lceil \delta_1 n \rceil$ coefficients in $\{\pm 1\}$, $d_2 = \lceil \delta_2 n \rceil$ coefficients in $\{\pm 2\}$ and all other coefficients set to 0 until $\mathbf{f}$ is invertible.[4] The secret key is given by $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^t = (\mathbf{f}, 2\mathbf{g} + 1)^t$.

The public key is then computed as follows: set $\mathbf{a}_q = (2\mathbf{g}+1)/\mathbf{f} \in R_q$ ($\mathbf{a}_q$ is defined as a quotient *modulo* q). Next, define $\mathbf{A} = (2\mathbf{a_q}, q - 2) \in R_{2q}^{1\times 2}$. One easily verifies that:

$$\begin{aligned} \mathbf{AS} &= 2\mathbf{a}_q \cdot \mathbf{f} - 2(2\mathbf{g} + 1) = 0 && \bmod q \\ \mathbf{AS} &= q(2\mathbf{g} + 1) = q \cdot 1 = 1 && \bmod 2\,, \end{aligned}$$

that is $\mathbf{AS} = q \bmod 2q$. Finally, $(\mathbf{A}, \mathbf{S})$ is a valid key pair for our scheme.

Denote by $\mathcal{K}_{n,\delta_1,\delta_2}$ the distribution that picks small $\mathbf{f}$ and $\mathbf{g}$ as uniform polynomials with exactly d_1 entries in $\{\pm 1\}$ and d_2 entries in $\{\pm 2\}$ and outputs the public key $\mathbf{B} = (\mathbf{a}, -1) \in R_q^{1\times 2}$ for $\mathbf{a} = (2\mathbf{g} + 1)/\mathbf{f}$.

The public key generated above $\mathbf{A}$ taken modulo q follows the distribution $2 \cdot \mathcal{K}_{n,\delta_1,\delta_2}$; that is, such key-pair generation algorithm gives a scheme based on $R\text{-SIS}^{\mathcal{K}_{n,\delta_1,\delta_2}}_{q,1,2,\beta}$ by Theorem 5.3.

Working with $\mathbf{A}$ *in Hermite Normal Form.* To compress our signature in Section 6.3.6 we need to have $\mathbf{A}$ in Hermite Normal Form (as in [GLP12]). Now, during the key-generation process, we explicitly constructed $\mathbf{A} = (\mathbf{a}_1, q - 2)$ such that

$$\mathbf{a}_1 \cdot \mathbf{s}_1 + (q - 2)\mathbf{s}_2 = q \bmod 2q\,.$$

Let us define ζ such that $\zeta \cdot (q - 2) = 1 \bmod 2q$. Next, instead of calling the random oracle during the signing and verification processes on $(\mathbf{A}\mathbf{y} \bmod 2q, \mu)$, we call it on $\big((\zeta\mathbf{A})\mathbf{y} \bmod 2q, \mu\big)$, because $\zeta\mathbf{A} = (\zeta\mathbf{a}_1, 1)$ is in Hermite Normal Form. We defer to Section 6.3.7 for the detailed algorithms of our signature scheme. In the following we denote $\mathbf{A} = (\zeta \cdot \mathbf{a}_1, 1)$ the public key.

6.2.3 A Tighter Bound on $\|\mathbf{Sc}\|$

As mentioned in Chapter 5, the repetition rate during our signing process is given by

$$M = \exp\Big(\frac{1}{2\alpha^2}\Big)\,,$$

where α is such that $\sigma \geqslant \alpha \cdot \max_{\mathbf{S},\mathbf{c}} \|\mathbf{Sc}\|$. Therefore for a fixed repetition rate (*i.e.* a fixed α), the standard deviation σ is linear in the maximum norm of $\|\mathbf{Sc}\|$. As a consequence, to reduce

[4] In order to get a better entropy/length ratio, we include a few entries in $\{\pm 2\}$ in the secret key, increasing resistance against the Hybrid attack (see Section 6.4).

the signature size, we want a bound as tight as possible for $\|\mathbf{Sc}\|$. In [Lyu12], Lyubashevsky used $\|\mathbf{Sc}\| \leqslant \|\mathbf{c}\|_1 \cdot \|\mathbf{S}\| = \kappa\|\mathbf{S}\|$. One could also use $\|\mathbf{Sc}\| \leqslant s_1(\mathbf{S}) \cdot \|\mathbf{c}\| = s_1(\mathbf{S}) \cdot \sqrt{\kappa}$ where $s_1(\mathbf{S})$ is the singular norm of $\mathbf{S}$.

In this section, we introduce a new measure of $\mathbf{S}$, adapted to the form of $\mathbf{c}$, which helps us achieve a tighter bound than with previous methods. We believe that this norm and the technique for bounding it could be of independent interest.

Definition 6.2. For any integer κ, we define $N_\kappa \colon \mathbb{R}^{m\times n} \to \mathbb{R}$ as:

$$N_\kappa(\mathbf{X}) = \max_{\substack{I\subset\{1,\dots,n\}\\ \#I=\kappa}} \sum_{i\in I} \left(\max_{\substack{J\subset\{1,\dots,n\}\\ \#J=\kappa}} \sum_{j\in J} T_{i,j} \right) \quad \text{where } \mathbf{T} = \mathbf{X}^t \cdot \mathbf{X} \in \mathbb{R}^{n\times n} .$$

The following proposition states that $\sqrt{N_\kappa(\mathbf{S})}$ is also an upper bound for $\|\mathbf{Sc}\|$.

Proposition 6.3. *Let $\mathbf{S} \in \mathbb{R}^{m\times n}$ be a real matrix. For any $\mathbf{c} \in \mathbb{B}^n_\kappa$, we have $\|\mathbf{Sc}\|^2 \leqslant N_\kappa(\mathbf{S})$.*

Proof. Set $I = J = \{i \in \{1,\dots,n\} : \mathbf{c}_i = 1\}$, which implies $\#I = \#J = \kappa$. Rewriting $\|\mathbf{S}\cdot\mathbf{c}\|^2 = \mathbf{c}^t \cdot \mathbf{S}^t \cdot \mathbf{S} \cdot \mathbf{c} = \mathbf{c}^t \cdot \mathbf{T} \cdot \mathbf{c} = \sum_{i\in I}\sum_{j\in J} T_{i,j}$, we can conclude from the definition of N_κ. □

In practice, we will use this upper bound in our implementation to bound $\|\mathbf{Sc}\|$ and derive the parameters. Some secret keys $\mathbf{S}$ will be rejected according to the value of $N_\kappa(\mathbf{S})$, which is easily computable. In addition to the gain from the use of bimodal Gaussians, this new upper bound lowers the standard deviation σ by a factor $\approx \sqrt{\kappa}/2$ compared to [Lyu12].

Computation of $N_\kappa(\mathbf{S})$. Recall that

$$N_\kappa(\mathbf{S}) = \max_{\substack{I\subset\{1,\dots,n\}\\ \#I=\kappa}} \sum_{i\in I} \left(\max_{\substack{J\subset\{1,\dots,n\}\\ \#J=\kappa}} \sum_{j\in J} T_{i,j} \right) \quad \text{where } \mathbf{T} = \mathbf{S}^t \cdot \mathbf{S} \in \mathbb{R}^{n\times n} .$$

In order to obtain $N_\kappa(\mathbf{S})$, note that it is not required to compute the $2\cdot\binom{n}{\kappa}$ sums in the definition. Indeed, it suffices to compute $\mathbf{T} = \mathbf{S}^t \cdot \mathbf{S}$, then sort the columns of $\mathbf{T}$, sum the κ larger values in each line, sort the resulting vector and to sum its κ larger components. Notice moreover that working in $R_{2q} = \mathbb{Z}_{2q}[x]/(x^n+1)$ implies that $\mathbf{S}$ is composed of rotations (possibly with opposed coefficients) of $\mathbf{s}_i$'s, and this (ideal) structure is thus also present in $\mathbf{T}$. Thus it suffices to compute the vector

$$\mathbf{t} = \left(\langle \mathbf{s}_1, \mathbf{s}_1\rangle + \langle \mathbf{s}_2, \mathbf{s}_2\rangle, \langle \mathbf{s}_1, x\cdot\mathbf{s}_1\rangle + \langle \mathbf{s}_2, x\cdot\mathbf{s}_2\rangle, \dots, \langle \mathbf{s}_1, x^{n-1}\cdot\mathbf{s}_1\rangle + \langle \mathbf{s}_2, x^{n-1}\cdot\mathbf{s}_2\rangle\right) ,$$

and derive $\mathbf{T} = (\mathbf{t}, x\cdot\mathbf{t}, \dots, x^{n-1}\cdot\mathbf{t})^t$.

Theoretical Bound. We provide below a (theoretical) asymptotic bound on $N_\kappa(\mathbf{S})$ for completeness. The following proposition easily generalizes to the form of our secret keys (see Corollary 6.6).

Proposition 6.4. *For a fixed density $\delta \in (0,1)$, and $w = \lceil \delta n \rceil$, let $\mathbf{s} \in \mathbb{Z}[x]/(x^n+1)$ be chosen uniformly in $\mathbb{T}^n_w$, and $\mathbf{S} \in \mathbb{Z}^{n\times n}$ denote its matrix representation. Then, for any $\epsilon > 0$, we have:*

$$N_\kappa(\mathbf{S}) \leqslant w\kappa + \kappa^2 \mathcal{O}\left(w^{1/2+\epsilon}\right)$$

except with negligible probability.

Proof. The first term $w\kappa$ arises from the diagonal coefficients of $\mathbf{T} = \mathbf{S}^t\cdot\mathbf{S}$, equals to $\|\mathbf{s}\|^2 = w$. It remains to bound the non-diagonal terms of $\mathbf{T}$. For $i \neq j$,

$$Y_{i,j} = \sum_{1\leqslant k\leqslant n} \varepsilon_{i,j,k} \cdot s_{i+k} \cdot s_{j+k} ,$$

where $\varepsilon_{i,j,k} \in \{\pm 1\}$ are some fixed coefficients, and the indices are taken modulo n. The key argument is to split this sum into two parts, so that each part contains only independent terms.

This is possible when $i - j \neq 0$ and n is a power of 2: one easily checks that there exists a set $K \subset \mathbb{Z}_n$ such that $K + i$ and $K + j$ form a partition of $\mathbb{Z}_n$. Thus, we rewrite

$$Y_{i,j} = \sigma_{i,j} + \bar{\sigma}_{i,j} \quad \text{where } \sigma_{i,j} = \sum_{k \in K} \varepsilon_{i,j,k} \cdot s_{i+k} \cdot s_{j+k} \text{ and } \bar{\sigma}_{i,j} = \sum_{k \in \mathbb{Z}_n \setminus K} \varepsilon_{i,j,k} \cdot s_{i+k} \cdot s_{j+k} \ .$$

Focusing on the sum $\sigma_{i,j}$ (a similar argument holds for $\bar{\sigma}_{i,j}$), one can restrict the sum to its non-zero terms and note that the remaining terms are uniformly random in $\{-1, 1\}$ and independent from each other. Finally $\sigma_{i,j}$ is the sum of at most w uniform variables over $\{-1, 1\}$ and therefore $\sigma_{i,j} \leqslant w^{1/2+\epsilon}$ except with negligible probability.[5] □

Remark 6.5. For the sake of simplicity, we denote $N_\kappa(\mathbf{v}) = N_\kappa(\mathbf{V})$ where $\mathbf{V} \in \mathbb{Z}^{n \times n}$ is the matrix representation of $\mathbf{v} \in \mathbb{Z}[x]/(x^n + 1)$.

Corollary 6.6. *Let $\mathbf{f}, \mathbf{g} \in \mathbb{Z}[x]/(x^n + 1)$ be chosen uniformly in $\mathbb{T}_w^n$, $\mathbf{F}, \mathbf{G} \in \mathbb{Z}^{n \times n}$ be their matrix representations, and set $\mathbf{S}^t = (\mathbf{F}|2\mathbf{G} + \mathbf{I}_n) \in \mathbb{Z}^{n \times 2n}$. Then,*

$$N_\kappa(\mathbf{S}) \leqslant (5w + 1)\kappa + \kappa^2 \mathcal{O}\left(w^{1/2+\epsilon}\right) \ .$$

Proof. This follows easily from the fact that $\mathbf{S}^t \cdot \mathbf{S} = \mathbf{F}^t \cdot \mathbf{F} + 4\mathbf{G}^t \cdot \mathbf{G} + \mathbf{G} + \mathbf{G}^t + \mathbf{I}_n$, yielding $N_\kappa(\mathbf{S}) \leqslant N_\kappa(\mathbf{F}) + 4N_\kappa(\mathbf{G}) + 2\kappa^2 + \kappa$. □

Rejection According to $N_\kappa(\mathbf{S})$. In practice after generating $\mathbf{S}$, we restart when $N_\kappa(\mathbf{S}) \geqslant C^2 \cdot 5 \cdot (d_1 + 4d_2) \cdot \kappa$ for a fixed constant C. This constant is chosen so that 25% of the keys are accepted, decreasing the overall security by at most 2 bits.

6.2.4 Final KeyGen Algorithm

Algorithm 6.1 BLISS Key Generation.

1: **function** KeyGen(Parameters $\delta_1, \delta_2, \kappa, q, n$)
2: Choose $\mathbf{f}, \mathbf{g}$ as uniform polynomials with exactly $d_1 = \lceil \delta_1 n \rceil$ entries in $\{\pm 1\}$ and $d_2 = \lceil \delta_2 n \rceil$ entries in $\{\pm 2\}$
3: $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^t \leftarrow (\mathbf{f}, 2\mathbf{g} + 1)^t \in R_{2q}^{2\times 1}$
4: **if** $N_\kappa(\mathbf{S}) \geqslant C^2 \cdot 5 \cdot (d_1 + 4d_2) \cdot \kappa$ **then**
5: **restart** ▷ Restart if bound on $\|\mathbf{Sc}\|$ is too large
6: **end if**
7: $\mathbf{a}_q = (2\mathbf{g} + 1)/\mathbf{f} \bmod q$ (**restart** if $\mathbf{f}$ is not invertible)
8: $\zeta = (q - 2)^{-1} \bmod 2q$
9: **return** $(\mathbf{A}, \mathbf{S})$ where $\mathbf{A} = (\zeta 2\mathbf{a}_q, 1) \bmod 2q$
10: **end function**

6.3 Implementation Details

In this section, we present optimizations and discuss implementation issues for each step of the signing algorithm. To simplify the reader experience, let us recall the signing algorithm of BLISS (Algorithm 5.1, page 46):

[5] By Hoeffding bound for example, or classical properties of random walks.

Algorithm 6.2 Signature Algorithm.

1: **function** Sign(Message digest μ, public key $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2) \in \mathbb{Z}_{2q}^{1\times 2}$, secret key $\mathbf{S} \in \mathbb{Z}_{2q}^{2\times 1}$, std. dev. $\sigma \in \mathbb{R}$)
2: $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)^t \leftarrow (D_\sigma^n)^2$
3: $\mathbf{c} \leftarrow H(\mathbf{Ay} \bmod 2q, \mu)$ ▷ Compute the challenge
4: $b \leftarrow \{0, 1\}$
5: $\mathbf{z} \leftarrow \mathbf{y} + (-1)^b \mathbf{Sc}$
6: $b \leftarrow \mathcal{B}_{\Big/\left(M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc}\rangle}{\sigma^2}\right)\right)}^{1}$
7: **if** $b = 1$ **then** ▷ output with probability $\frac{1}{M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc}\rangle}{\sigma^2}\right)}$
8: **return** $(\mathbf{z}, \mathbf{c})$
9: **else**
10: **restart**
11: **end if**
12: **end function**

6.3.1 Multiplication of two polynomials

In Line 3 of Algorithm 6.2, the element $\mathbf{Ay} = \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{a}_2 \cdot \mathbf{y}_2 \in \mathbb{Z}_{2q}[x]/(x^n + 1)$ is given as input to the random oracle. Since $\mathbf{a}_2 = 1$ is a constant, $\mathbf{a}_2 \cdot \mathbf{y}_2$ is straightforward to obtain. It remains to (efficiently) compute the product of $\mathbf{a}_1$ by $\mathbf{y}_1$ over $\mathbb{Z}_{2q}[x]/(x^n + 1)$.

Because of the particular shape of $\mathbf{a}_1$ in the NTRU-like key generation, namely that $\mathbf{a}_1$ is lifted from $\mathbb{Z}_q[x]/(x^n + 1)$ to $\mathbb{Z}_{2q}[x]/(x^n + 1)$ by multiplying its coefficients by $2 \cdot \zeta$ (*i.e.* $\mathbf{a}_1 = 2 \cdot \zeta \cdot \mathbf{a'}_1$), computing $\mathbf{a_1} \cdot \mathbf{y}_1$ over $\mathbb{Z}_{2q}[x]/(x^n + 1)$ can be done by computing the product $\mathbf{a'}_1 \cdot \mathbf{y}_1$ over $\mathbb{Z}_q[x]/(x^n + 1)$ and then multiplying the coefficients of the result by $2 \cdot \zeta$. Now multiplying two polynomials of $\mathbb{Z}_q[x]/(x^n + 1)$ for q prime is made efficient by choosing a modulus q such that $q = 1 \bmod 2n$ (then there exists a primitive $2n$-th root ω of unity modulo q). Finally, the multiplication can be done in complexity $\mathcal{O}(n \log n)$ via a Number Theoretic Transform (*i.e.* Fast Fourier Transform over a finite field). Details on these standard techniques can be found for example in [Ber08, PG12]. Note that one does not need to work with vectors of size $2n$ as the component-wise multiplication of the NTT representations of size n of $\mathbf{a'}_1(\omega x)$ and $\mathbf{y}_1(\omega x)$ gives the NTT representation of $[\mathbf{a'}_1 \cdot \mathbf{y}_1](\omega x) \in \mathbb{Z}_q[x]/(x^n + 1)$.

6.3.2 Multiplication of S by a sparse vector c

In Line 5 of Algorithm 6.2, one should compute $\mathbf{Sc}$. Let $\mathbf{S}_i$, $i = 1, 2$ denotes the $n \times n$ matrix over $\mathbb{Z}_{2q}$ whose columns vectors are the $x^j \cdot \mathbf{s}_i$'s for $j = 0, \ldots n - 1$ In particular we have that

$$\mathbf{s}_i \cdot \mathbf{c} = \mathbf{S}_i \mathbf{c} \, .$$

Now, since $\mathbf{c}$ is a sparse binary vector, one should not use the NTT to compute $\mathbf{s}_i \cdot \mathbf{c}$ for this step (contrary to Section 6.3.1). Indeed, the absolute value of the coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ is smaller than 5, yielding $\|\mathbf{s}_i \cdot \mathbf{c}\|_\infty \leqslant 5\kappa \ll 2q$, $i = 1, 2$. Therefore, computing $(\mathbf{s}_1 \cdot \mathbf{c})$ and $(\mathbf{s}_2 \cdot \mathbf{c})$ can be performed very efficiently by additions over $\mathbb{Z}$ (*i.e.* without reduction modulo $2q$) of κ pre-stored columns of $\mathbf{S}_i$ (in general). Notice moreover that working over $R_{2q} = \mathbb{Z}_{2q}[x]/(x^n + 1)$ allows to reduce the memory storage overhead to zero: all the columns of $\mathbf{S}_i$ are rotations (possibly with opposite coefficients) of $\mathbf{s}_i$.

6.3.3 Hashing to $\mathbb{B}_\kappa^n$

We discuss how to build a hash function outputting uniform vectors in $\mathbb{B}_\kappa^n$ from a standard hash function H (used in Line 3 of Algorithm 6.2). In [Lyu12], it was suggested to use a Hash function with $\kappa \log_2 \binom{n}{\kappa}$ bits of output (recall that $\#\mathbb{B}_\kappa^n = \binom{n}{\kappa}$, and thus $\#\mathbb{T}_\kappa^n = 2^\kappa \binom{n}{\kappa}$), and then apply a one-to-one map to $\mathbb{T}_\kappa^n$. Such a mapping can be found in [FS96] but its complexity is quadratic in n; this is quite inefficient especially for large parameters. To avoid this costly algorithm, the authors

of [GLP12] used an efficient procedure injectively mapping 160-bit strings to $\mathbb{T}_{32}^{512}$; they increased the value κ from 20 to 32 to gain efficiency, yielding a larger signature size.

Overview. We here give an alternative solution that is both efficient and optimal (*i.e.* κ is minimal for a target entropy) to produce random elements in $\mathbb{B}_\kappa^n$. In a few words, our approach consists in obtaining $\kappa' > \kappa$ values $x_1 \dots x_{\kappa'}$ in $\mathbb{Z}_n$, and setting the coordinates $\mathbf{c}_{x_i}$ of the challenge $\mathbf{c}$ to 1, starting from $i = 1$, and until $\|\mathbf{c}\|_1 = \kappa$. If some coordinate $\mathbf{c}_{x_j}$ is already set to 1 one just ignores this x_j. If we run out of values x_j, we would restart the process using a different seed. In the following we describe more precisely this algorithm and show it indeed produces a uniform random function over $\mathbb{B}_\kappa^n$ if H is indeed a uniform random function over $\mathbb{Z}_n^k$.

Detailed Construction and Correctness. Let n be a power of 2 and $H_0 \colon \{0,1\}^* \to \mathbb{Z}_n^{\kappa'}$ with $\kappa' > \kappa$ be a random function outputting $\kappa' \log_2 n$ bits (parsed as κ' elements in $\mathbb{Z}_n$). We consider the set $S \subset \mathbb{Z}_n^{\kappa'}$ of vectors that have at least κ different entries. The probability that a uniform element in $\mathbb{Z}_n^{\kappa'}$ lies in S is:

$$A = 1 - \frac{|\mathbb{Z}_n^{\kappa'} \setminus S|}{|\mathbb{Z}_n^{\kappa'}|}.$$

When A is not negligible, one can efficiently build a random function $\widetilde{H} \colon \{0,1\}^* \to S$ as $\widetilde{H}(x) = H(x|i)$, where i is the smallest index such that $H(x|i) \in S$. This is somehow a rejection sampling technique applied to a random function. Finally, in average, one call of $\widetilde{H}$ requires $1/A$ calls to H.[6]

Lemma 6.7. *With the notation above,* $|\mathbb{Z}_n^{\kappa'} \setminus S| \leqslant \binom{n}{\kappa-1}(\kappa-1)^{\kappa'}$.

Proof. Note that $|\mathbb{Z}_n^{\kappa'} \setminus S|$ is the set of vectors over $\mathbb{Z}_n$ of length κ' with at most $\kappa - 1$ distinct coordinates. To obtain this set, one may first choose a subset $K \subset \mathbb{Z}_n$ of size $\kappa - 1$ ($\binom{n}{\kappa-1}$ choices), and then chooses the κ' coordinates in K ($(\kappa-1)^{\kappa'}$ choices). Note that vectors with strictly less than $\kappa - 1$ coordinates have been counted several times. More formally:

$$\mathbb{Z}_n^{\kappa'} \setminus S = \bigcup_{\substack{K \subset \mathbb{Z}_n \\ |K| < \kappa-1}} K^{\kappa'} = \bigcup_{\substack{K \subset \mathbb{Z}_n \\ |K| = \kappa-1}} K^{\kappa'}. \qquad \square$$

It remains to map the domain S to $\mathbb{B}_\kappa^n$. For $\mathbf{x} \in S$, let I be the set of the κ first distinct coordinates values of $\mathbf{x}$, and set $f(\mathbf{x}) = \sum_{i \in I} \mathbf{e}_i \in \mathbb{B}_\kappa^n$ where $\mathbf{e}_1, \dots, \mathbf{e}_n$ are the canonical vectors of $\mathbb{Z}^n$. Each image $\mathbf{y} \in \mathbb{B}_\kappa^n$ has the same amount of f-preimages in S, therefore $\widehat{H} \colon \{0,1\}^* \to \mathbb{B}_\kappa^n$ defined as $\widehat{H}(x) = f \circ \widetilde{H}(x)$ is also a random function.

6.3.4 Gaussian Sampling

In Line 2 of Algorithm 6.2, we want to produce $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)^t$ where $\mathbf{y}_1, \mathbf{y}_2$ are polynomials over $\mathbb{Z}_{2q}[x]/(x^n + 1)$ with coefficients distributed according to a centered discrete Gaussian distribution of standard deviation σ. In Chapter 4, we provide a new technique to *efficiently* perform discrete Gaussian sampling on constrained devices. However in an environment with enough memory, using the cumulative distribution table algorithm (the inversion algorithm) is the easiest and fastest solution (see Table 4.1). Namely, we tabulate the approximate cumulative distribution of the desired distribution, *i.e.* the probabilities $p_z = \Pr[x \leqslant z : x \leftarrow D_\sigma]$ for $z \in [-\tau\sigma, \tau\sigma]$, precomputed with λ' bits of precision, where $\tau = 13.4$.[7] At sampling time, one generates $y \in [0, 1)$ uniformly at random, then performs a binary search through the table to locate some $z \in \mathbb{Z}$ such that $y \in [p_{z-1}, p_z)$ and outputs z. A `gcc`-profiling of our program [DL13] reveals that this step takes about 35% of the entire running-time, including the entropy generation using `sha-512`.

[6] For our parameters, this gives $1/A \leqslant 1.00001$ using a 512-bit hash function for H (*e.g.* BLISS-IV: $n = 512$, $\kappa' = 56$, $\kappa = 39$).

[7] The number of precision bits λ' has to be selected according to an upper bound on the number of signatures. More precisely, if an attacker can obtain an arbitrary number N of signatures, we want the distribution *of the tuple* of all these signatures to be statistically close to the good distribution. Since each signature contains $2n$ Gaussian values, we have $2nN$ Gaussian values in total and this implies that we need to set $\lambda' \geqslant \lambda + \log_2(2nN)$.

6.3.5 Rejection Sampling according to $1/\exp$ and $1/\cosh$

In Lines 6-8 of Algorithm 5.1, one should reject with probability

$$1/(M \exp(-x/f) \cosh(x'/f)) \ .$$

To avoid floating-point computations of the transcendental functions exp and cosh, we use the techniques described in Section 4.3 of Chapter 4 to do it efficiently with a very small memory footprint. (Note that when working on a constrained device, the precomputed values can be the same both for Gaussian sampling and this rejection sampling step.)

6.3.6 Signature Compression

Similarly to [GLP12], we would like to reduce our signature size by compressing it. Recall that a signature is a pair $(\mathbf{z}, \mathbf{c})$ where $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2)^t$ follows the Gaussian distribution D_σ^{2n}. In [GLP12], the signature size is reduced by dropping almost all the information about $\mathbf{z}_2$ in the signature. Such a strategy impacts security, as it reduces to an easier SIS problem (it may allow an attacker to forge using longer vectors). Let us describe a similar feature for our signature scheme.

Dropping the low-order bits of $\mathbf{z_2}$. We denote by d the number of bits we would like to drop in $\mathbf{z}_2$. For every integer x in the range $[-q, q)$ and any positive integer d, x can be uniquely written

$$x = \lfloor x \rceil_d \cdot 2^d + [x \bmod 2^d] \ ,$$

where $[x \bmod 2^d] \in [-2^{d-1}, 2^{d-1})$. Thus $\lfloor x \rceil_d$ can be viewed as the "high-order bits" of x and $[x \bmod 2^d]$ as its "low-order bits".

Recall that the random oracle H input is

$$\begin{aligned} \big(\lfloor \mathbf{Ay} \rceil_d, \mu\big) &= \big(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q \rceil_d, \mu\big) \\ &= \big(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} + \mathbf{z}_2 \bmod 2q \rceil_d, \mu\big). \end{aligned} \tag{6.1}$$

The idea of [GLP12], transposed to our settings, was to define a vector $\widetilde{\mathbf{z}_2}$ with coefficients in $\{0, \pm 2^d\}$ and a limited number of coefficients $\widetilde{\mathbf{z}_2}[i] = \mathbf{z}_2[i]$ (coming from the need of reduction modulo $2q$ after the addition with small but non negligible probability) such that

$$\lfloor \mathbf{Ay} \rceil_d = \lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} + \widetilde{\mathbf{z}_2} \bmod 2q \rceil_d \ .$$

Unfortunately the workaround which consists in storing some coefficients uncompressed, *i.e.* of the form $\widetilde{\mathbf{z}_2}[i] = \mathbf{z}_2[i]$, yields a signature scheme which is not strongly unforgeable. Indeed it is easy to forge a signature by modifying the least-significant bit of one of the uncompressed values, and this does not modify the high-order bits of the sum with very high probability.[8]

Let us describe how to solve this issue for our signature scheme. We want to replace $\mathbf{z}_2$ by a small vector $\mathbf{z}_2^\dagger$ such that

$$\lfloor \mathbf{Ay} \rceil_d = \lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \bmod 2q \rceil_d + \mathbf{z}_2^\dagger \ .$$

Unfortunately without additional modification, the security proof does not go through because of a similar issue as in [GLP12], *i.e.* the coefficients $\mathbf{z}_2[i]$ of $\mathbf{z}_2$ which, added to $\big(\zeta \cdot (\mathbf{a}_1 \cdot \mathbf{z}_1)[i] + \zeta \cdot q \cdot \mathbf{c}[i]\big)$, force us to reduce the result modulo $2q$ in Equation (6.1). Let us define $p = \lfloor 2q/2^d \rfloor$; we have $2q = p \cdot 2^d + \nu$ with a small ν (typically $\nu = 1$ in our parameters). Now we modify the random oracle H input by

$$\big(\lfloor \mathbf{Ay} \rceil_d \bmod p, \mu\big) \ ,$$

and define

$$\mathbf{z}_2^\dagger = \big(\lfloor \mathbf{Ay} \rceil_d - \lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \bmod 2q \rceil_d \bmod p\big) \in [0, p)^n \ .$$

[8] As a direct consequence, the scheme of [GLP12] is not strongly unforgeable.

The coefficients of $\mathbf{z}_2^\dagger$ are small modulo p. We redefine the signature to be $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$ instead of $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$, and during the verification, we check that

$$H\left(\mathbf{z}_2^\dagger + \lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \bmod 2q \rceil_d \bmod p, \mu\right) = \mathbf{c} \ ,$$

that $\|(\mathbf{z}_1, 2^d \mathbf{z}_2^\dagger)^t\| \leqslant B_2$ and that $\|(\mathbf{z}_1, 2^d \mathbf{z}_2^\dagger)^t\|_\infty \leqslant B_\infty$.

Finally, we have the following theorem:

Theorem 6.8. *Let us consider the signature scheme of Section 6.3.7. Assume that $d \geqslant 3$, $q \equiv 1 \bmod 2^{d-1}$, and $2B_\infty + (2^d + 1) < q/2$. Suppose there is a polynomial-time algorithm $\mathcal{F}$ which succeeds in forging with non negligible probability. Then there exists a polynomial-time algorithm which can solve the R-SIS$^{\mathcal{K}_{n,\delta_1,\delta_2}}_{q,n,m,\beta}$ problem for $\beta = 2B_2 + (2^d + 1)\sqrt{n}$.*

Proof. The proof of this theorem follows the same blueprint than the proof of Theorem 5.3. Namely, by a straightforward adaptation of Lemma 5.4, one can show that our signing algorithm can be replaced by Hybrid 3 (Algorithm 6.3) restarted if necessary. Next, an adaptation of Lemma 5.5 states that if an algorithm can produce a forgery with non-negligible probability when the signing algorithm is replaced by Hybrid 3 (restarted), then we can use it to recover a vector $\mathbf{v} \neq \mathbf{0} \bmod q$ such that $\|\mathbf{v}\| \leqslant \beta = 2B_2 + (2^d + 1)\sqrt{n}$ and $\mathbf{A}\mathbf{v} = \mathbf{0} \bmod q$.

Algorithm 6.3 HYBRID$_3$

function HYBRID$_3$(Message digest μ, public key $\mathbf{A} \in R_{2q}^2$, std. dev. $\sigma \in \mathbb{R}$)
  $\mathbf{c} \leftarrow \mathbb{B}_\kappa^n$
  $\mathbf{z}_1, \mathbf{z}_2 \leftarrow \mathcal{D}_\sigma^n$
  $b \leftarrow \mathcal{B}_{1/M}$
  if $b = 1$ **then** ▷ output with probability $1/M$
    program $H(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} + \mathbf{z}_2 \rceil_d \bmod p, \mu) = \mathbf{c}$
    $\mathbf{z}_2^\dagger \leftarrow (\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} + \mathbf{z}_2 \rceil_d - \lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \rceil_d) \bmod p$
    return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
  end if
end function

In what follows, we focus on the modifications in the proof of Lemma 5.5 to deal with the dropping bits, *i.e.* we assume that $\mathcal{F}$ succeeds in forging the signature by outputting $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$ where $\mathbf{c} = \mathbf{c}_j \in \{\mathbf{c}_1, \ldots, \mathbf{c}_t\}$ was obtained from either a previous signing query, or a previous random oracle query.

We have the following preliminaries facts:

Lemma 6.9. *Let q be an odd integer and define $\zeta \in [0, 2q-1]$ such that $\zeta \cdot (q-2) = 1 \bmod 2q$. Then $\zeta = \frac{q-1}{2}$ if $(q-1)/2$ is odd or $\zeta = \frac{q-1}{2} + q$ if $(q-1)/2$ is even.*

Proof. We have that

$$\frac{q-1}{2} \cdot (q-2) = q \cdot \frac{q-1}{2} - q + 1 = 1 \bmod q \ .$$

Therefore $\zeta = \frac{q-1}{2} \bmod q$ and the lemma holds according to the parity of $(q-1)/2$. □

Lemma 6.10. *Let $d \geqslant 2$, q be an integer such that $q \equiv 1 \bmod 2^{d-1}$, and let $p = \lfloor 2q/2^d \rfloor$. Then $p \cdot 2^d = 2q - 2$.*

Assume the challenger has a signature $(\mathbf{z'}_1, \mathbf{z'}_2^\dagger, \mathbf{c'}_j)$ such that

$$\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c}_j \rceil_d + \mathbf{z}_2^\dagger \bmod p = \lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z'}_1 + \zeta \cdot q \cdot \mathbf{c'}_j \rceil_d + \mathbf{z'}_2^\dagger \bmod p \ .$$

There exists $\mathbf{k} \in \{0, \pm 1\}^n$ such that the following equation holds over $\mathbb{Z}$:

$$\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c}_j \rceil_d - \lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z'}_1 + \zeta \cdot q \cdot \mathbf{c'}_j \rceil_d + \mathbf{z}_2^\dagger - \mathbf{z'}_2^\dagger = \mathbf{k}p \ .$$

Now we multiply the previous equation by 2^d, and this yields modulo $2q$:

$$\zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c}_j - \mathbf{e} - \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z'}_1 - \zeta \cdot q \cdot \mathbf{c'}_j + \mathbf{e'} + 2^d(\mathbf{z}_2^\dagger - \mathbf{z'}_2^\dagger) = \mathbf{k} \cdot p2^d \bmod 2q \ ,$$

where $\mathbf{e} = [\zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c}_j \bmod 2q] \bmod 2^d$ and $\mathbf{e'} = [\zeta \cdot \mathbf{a}_1 \cdot \mathbf{z'}_1 + \zeta \cdot q \cdot \mathbf{c'}_j \bmod 2q] \bmod 2^d$. This yields by Lemma 6.10:

$$(\zeta \cdot \mathbf{a}_1) \cdot (\mathbf{z}_1 - \mathbf{z'}_1) + 2^d(\mathbf{z}_2^\dagger - \mathbf{z'}_2^\dagger) + \zeta \cdot q \cdot (\mathbf{c}_j - \mathbf{c'}_j) + (\mathbf{e'} - \mathbf{e}) + 2\mathbf{k} = \mathbf{0} \bmod 2q \ . \tag{6.2}$$

Thus, if we define

$$\mathbf{v} = \big(\mathbf{z}_1 - \mathbf{z'}_1, 2^d(\mathbf{z}_2^\dagger - \mathbf{z'}_2^\dagger) + (\mathbf{e'} - \mathbf{e}) + 2\mathbf{k}\big)^t \in (\mathbb{Z}[x]/(x^n+1))^2 \ ,$$

we have that

$$(\zeta \cdot \mathbf{a}_1, 1) \cdot \mathbf{v} = \mathbf{0} \bmod q \ ,$$

and thus multiplying by 2:

$$(\mathbf{a}_1, 2) \cdot \mathbf{v} = \mathbf{0} \bmod q \ .$$

Now, we have that $\|\mathbf{v}\|_2 \leqslant 2B_2 + (2^d+1) \cdot \sqrt{n}$ and $\|\mathbf{v}\|_\infty \leqslant 2B_\infty + (2^d+1) < q/2$. Indeed

$$\begin{aligned}
\|\mathbf{v}\|_2 &\leqslant \|\big(\mathbf{z}_1 - \mathbf{z'}_1, 2^d(\mathbf{z}_2^\dagger - \mathbf{z'}_2^\dagger)\big)^t\|_2 + \|\big(\mathbf{0}, (\mathbf{e'} - \mathbf{e} + 2\mathbf{k})\big)^t\|_2 \\
&\leqslant 2B_2 + \|\big(\mathbf{0}, (\mathbf{e'} - \mathbf{e} + 2\mathbf{k})\big)^t\|_\infty \cdot \sqrt{n} \\
&\leqslant 2B_2 + (\|\big(\mathbf{0}, (\mathbf{e'} - \mathbf{e})\big)^t\|_\infty + 2\|\mathbf{k}\|_\infty) \cdot \sqrt{n} \\
&\leqslant 2B_2 + (2^d - 1 + 2) \cdot \sqrt{n} \\
&\leqslant 2B_2 + (2^d + 1) \cdot \sqrt{n} \ .
\end{aligned}$$

Similarly for the infinite norm, we get

$$\|\mathbf{v}\|_\infty \leqslant 2B_\infty + (2^d+1) < q/2 \ .$$

It remains to show that $\mathbf{v} \neq \mathbf{0} \bmod q$ to conclude. By the condition $\|\mathbf{v}\|_\infty < q/2$, it suffices to show that $\mathbf{v} \neq \mathbf{0} \bmod 2q$.

Case #1. **[$\mathbf{z}_1 \neq \mathbf{z}'_1 \bmod 2q$].** Since

$$\mathbf{v} = \big(\mathbf{z}_1 - \mathbf{z'}_1, 2^d(\mathbf{z}_2^\dagger - \mathbf{z'}_2^\dagger) + (\mathbf{e'} - \mathbf{e}) + 2\mathbf{k}\big)^t \ ,$$

we have $\mathbf{v} \neq \mathbf{0} \bmod 2q$. This case includes both type-1 and type-2 forgeries.

Case #2. **[$\mathbf{z}_1 = \mathbf{z}'_1 \bmod 2q$ and $\mathbf{c}_j = \mathbf{c'}_j$].** In that case, we have $\mathbf{e} = \mathbf{e'}$, and for the signatures to be different we have $\mathbf{z}_2^\dagger \neq \mathbf{z'}_2^\dagger$. Therefore

$$\mathbf{v} = \big(\mathbf{0}, 2^d(\mathbf{z}_2^\dagger - \mathbf{z'}_2^\dagger) + 2\mathbf{k}\big)^t \ .$$

Now $\|2\mathbf{k}\|_\infty < 2^d$, then $\mathbf{v} \neq \mathbf{0} \bmod 2q$. This case is only possible for type-1 forgeries.

Case #3. **[$\mathbf{z}_1 = \mathbf{z}'_1 \bmod 2q$, $\mathbf{c}_j \neq \mathbf{c'}_j$ and $\mathbf{z}_2^\dagger = \mathbf{z'}_2^\dagger \bmod 2q$].** In that case, Equation (6.2) yields

$$\mathbf{e'} - \mathbf{e} + 2\mathbf{k} = \zeta \cdot q \cdot (\mathbf{c}_j - \mathbf{c'}_j) \bmod 2q \ .$$

Now $\mathbf{c}_j - \mathbf{c'}_j \neq \mathbf{0} \bmod 2$, therefore $\mathbf{e'} - \mathbf{e} + 2\mathbf{k} \neq \mathbf{0} \bmod 2q$. Since

$$\mathbf{v} = \big(\mathbf{0}, (\mathbf{e'} - \mathbf{e}) + 2\mathbf{k}\big)^t \ ,$$

we have $\mathbf{v} \neq \mathbf{0} \bmod 2q$. This case is only possible for type-2 forgeries.

Case #4. $[\mathbf{z}_1 = \mathbf{z}'_1 \bmod 2q$**,** $\mathbf{c}_j \neq \mathbf{c}'_j$ **and** $\mathbf{z}_2^\dagger \neq \mathbf{z}'^\dagger_2 \bmod 2q]$**.** In that case

$$\mathbf{v} = \left(\mathbf{0}, 2^d(\mathbf{z}_2^\dagger - \mathbf{z}'^\dagger_2) + (\mathbf{e}' - \mathbf{e}) + 2\mathbf{k}\right)^t .$$

Since $\mathbf{c}_j \neq \mathbf{c}'_j$, there exists i such that $\mathbf{c}_j[i] \neq \mathbf{c}'_j[i]$. Without loss of generality, we can assume that $\mathbf{c}'_j[i] = 1$ and thus $\mathbf{c}_j[i] = 0$. Therefore,

$$\mathbf{e}'[i] = \left(x + \zeta \cdot q \bmod 2q\right) \bmod 2^d ,$$

and

$$\mathbf{e}[i] = x \bmod 2^d ,$$

where $x = \left(\zeta \cdot (\mathbf{a}_1 \cdot \mathbf{z}_1)[i]\right) \bmod 2q$. Now $\zeta \cdot q = q \bmod 2q$ because $\zeta = 1 \bmod 2$ by Fact 6.9. Therefore

$$\mathbf{e}'[i] = \left(x + q \bmod 2q\right) \bmod 2^d .$$

Now, $\left(x + q \bmod 2q\right) = x \pm q$ over $\mathbb{Z}$. Therefore,

$$\left(\mathbf{e}'[i] - \mathbf{e}[i]\right) \bmod 2^d = \left((x \pm q) - x\right) \bmod 2^d$$

is odd. This proves that $\mathbf{v}[i]$ is odd, and therefore that $\mathbf{v} \neq \mathbf{0} \bmod 2q$. This case is only possible for type-2 forgeries. □

Compressing Most Significant Bits of $\mathbf{z}_1$ and $\mathbf{z}_2^\dagger$. The simplest representation of the entries of $\mathbf{z}_1$ then requires $\lceil \log_2(8\sigma) \rceil \leqslant \log_2(16\sigma)$ bits. Yet, the entropy of these entries is actually smaller:

Lemma 6.11. *Let X be distributed as D_σ, that is a centered discrete Gaussian variable. Then the entropy of X is upper-bounded by:*

$$\mathcal{H}(X) \leqslant \frac{1}{\sigma^3} + \log_2(\sqrt{2\pi e}\sigma) \approx \log_2(4.1\sigma) .$$

Now, Huffman coding provides (almost) optimal encoding for data when their distribution is exactly known. More precisely:

Theorem 6.12 (Huffman Coding)**.** *For any random variable X over a finite support S, there exist an injective prefix-free code $C\colon S \to \{0,1\}^*$ such that:*

$$\mathcal{H}(X) \leqslant E\left[|C(X)|\right] < \mathcal{H}(X) + 1 .$$

To keep the compression efficient, we choose to only encode the highest bits of all entries; the lower are almost uniform and therefore we do not lose anything by not compressing them. Moreover, if by packing several independent variables $X_1, \ldots, X_k$, we can decrease the overhead to $1/k$.

6.3.7 Final Sign and Verify Algorithms

In this section, we describe the final algorithms to instantiate BLISS with the parameters of Section 6.5 (Algorithms 6.4 and 6.5). Note that to obtain the signature size indicated Table 6.6 (page 71), one need to use Huffman Coding to compress the highest bits of $\mathbf{z}_1$ and $\mathbf{z}_2^\dagger$. Let us define $p = \lfloor 2q/2^d \rfloor$ where d is the number of dropped bits.

6.4 Security Analysis

In this section, we describe how known attacks apply to our scheme. First, we describe in Section 6.4.1 combinatorial attacks on the secret key, namely brute-force and meet-in-middle attacks.

Then we consider lattice reduction attacks. Typical measurements of lattice problem hardness (the so called Hermite factor, see [GN08, CN11]) are given in Table 6.5 (page 70), measuring how hard it is to find vectors of a given norm in a random lattice. We first apply this measure to the hardness of the underlying SIS problem, as if the lattice used was truly random (*cf.* Section 6.4.2).

Algorithm 6.4 BLISS Signature Algorithm.

function Sign(Message μ, public key $\mathbf{A} = (\zeta \cdot \mathbf{a}_1, 1) \in \mathcal{R}_{2q}^{1\times 2}$, secret key $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^t \in \mathcal{R}_{2q}^{2\times 1}$)
  $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D_{\mathbb{Z}^n,\sigma}$
  $\mathbf{u} \leftarrow \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$
  $\mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$ ▷ Compute the challenge
  $b \leftarrow \mathcal{B}_{1/2}$ ▷ Choose a random bit
  $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
  $\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
  $b \leftarrow \mathcal{B}_{1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc}\rangle}{\sigma^2}\right)\right)}$
  if $b = 1$ **then** ▷ output with probability $\frac{1}{M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc}\rangle}{\sigma^2}\right)}$
    $\mathbf{z}_2^\dagger \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$
    return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
  else
    restart
  end if
end function

Algorithm 6.5 BLISS Verification Algorithm.

function Verify(Message μ, public key $\mathbf{A} = (\zeta \cdot \mathbf{a}_1, 1) \in \mathcal{R}_{2q}^{1\times 2}$, signature $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$)
  if $\|(\mathbf{z}_1, 2^d \cdot \mathbf{z}_2^\dagger)^t\|_2 > B_2$ or $\|(\mathbf{z}_1, 2^d \cdot \mathbf{z}_2^\dagger)^t\|_\infty > B_\infty$ **then**
    return Reject
  end if
  if $\mathbf{c} = H\left(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \rceil_d + \mathbf{z}_2^\dagger \bmod p, \mu\right)$ **then**
    return Accept
  else
    return Reject
  end if
end function

Yet, the lattice is not truly random, as by design it contains unusually short vectors. Therefore, one may try to directly recover the secret by lattice reduction: find the secret key $(\mathbf{f}, \mathbf{g})^t$ as a short vector in the primal lattice $\Lambda = \{(\mathbf{x}, \mathbf{y})^t \in R^2 : \mathbf{a}_q \mathbf{x} + \mathbf{y} = 0 \bmod q\}$. Unfortunately, the only study [GN08] of the behavior of lattice algorithms in the presence of unusually short vectors only considers the unique-SVP problem, in which there is only *one* unusually short vector. In the NTRU-like case, there is a basis of n of them. We provide new experiments showing that the behavior is similar; it is dictated by the ratio between the actual shortest vector, its expected length in a random lattice and the Hermite factor (*cf.* Section 6.4.3).

An alternative attack to recover unusually short vectors of a lattice is to use short (but quite larger) dual lattice $\Lambda^\times$ vectors to detect its presence, and then recover it [MR09, MM11] using search-to-decision reduction; quantification of this attack is detailed in Section 6.4.4.

Finally, it is possible to combine lattice reduction and combinatorial techniques: Howgrave-Graham designed in [HG07] an attack against NTRU keys combining a meet-in-the-middle strategy with lattice reduction. This attack applies to our scheme, as detailed in Section 6.4.5, but also on the previous related schemes [Lyu12, GLP12]. Note that there is no mention of this attack in the security analysis of the latter schemes; therefore in order to compare, we also include security measurements for those schemes.

We base our security projection on the BKZ-2.0 simulation methodology [CN11] that models the behavior of BKZ including several enhancements [Kan83, GNR10, HPS11].

Note that we only sketch the attack principles; we refer the interested reader to the original articles [HHGP+03, HG07, CN11, MR09, MM11] for more detail. We emphasize that the statistical attacks [NR09, MPSW09, DN12b] provably (*i.e.* information-theoretically) do not apply here because of rejection sampling: the output distribution of the signature scheme is independent of

the secret key.

6.4.1 Brute-force and Meet-in-the-Middle Key Recovery Attack

The key-recovery problem is as follows: given $\mathbf{a} \in \mathbb{Z}_q[x]/(x^n+1)$, find small polynomials $\mathbf{f}, \mathbf{g}$ such that $\mathbf{a}(2\mathbf{g}+1) - \mathbf{f} = \mathbf{0}$ (knowing that such a solution exists). Precisely, we know that both $\mathbf{f}$ and $\mathbf{g}$ have respectively $d_1 = \lceil n\delta_1 \rceil$ entries in $\{-1, +1\}$ and $d_2 = \lceil n\delta_2 \rceil$ entries in $\{-2, +2\}$.

Brute-force Key Recovery. The brute-force attack simply consists in picking a random vector $\mathbf{g}$ according to the key-generation distribution, and checking whether $\mathbf{f} = \mathbf{a}(2\mathbf{g}+1)$ is a polynomial with coefficients in $\{-2, -1, 0, 1, 2\}$. To measure the complexity of this attack, one simply measures the entropy of $\mathbf{g}$: this entropy yields a lower bound on the time to exhaust all possible values. The time complexity of this attack is therefore $T = 2^{d_1+d_2}\binom{n}{d_1}\binom{n-d_1}{d_2}$.

For more complex attacks, it may be simpler to model all the entries of the secret key as independent random variables, each of them having entropy:

$$e = \delta_0 \log_2 \delta_0 + \delta_1 \log_2 \frac{\delta_1}{2} + \delta_2 \log_2 \frac{\delta_2}{2}.$$

In this model, the total entropy is $n \cdot e$, which is at most $\log n$ greater than the true entropy.

Meet-in-the-Middle Attack. Odlyzko proposed a MiM attack of running time the square root of the latter attack (but with additional memory consumption). It was designed against the NTRU signature scheme, but also applies here. We refer to [HHGP+03] for details, and give only a short explanation of a simplified version: exhaust $\mathbf{g}_1$ as the first half bits of $\mathbf{g}$ and store $\mathbf{g}_1$ in several labeled boxes (of an hash table) according to the values of $\mathbf{f}_1 = \mathbf{a}(2\mathbf{g}_1 + 1)$. Then search for the second half $\mathbf{g}_2$ of $\mathbf{g}$ by computing $\mathbf{f}_2 = \mathbf{a}(2\mathbf{g}_2) \bmod q$: the labeling is designed so that to ensure a collision whenever $\mathbf{f}_1 + \mathbf{f}_2$ has its coefficients in $\{-2, -1, 0, 1, 2\}$.

This attacks runs in time and memory about $2^{n \cdot e/2}$, since the entropy of a half of the vector is $n \cdot e/2$.

6.4.2 Hardness of the underlying SIS problem

Attack Overview. In this section we measure the hardness of forging a signature according to our security proof. We will consider the running time necessary for the BKZ algorithm to find a vector of norm $\beta = 2B_2 + (2^d+1)\sqrt{n}$ in a random q-ary lattice according to the analysis of [CN11]. While the lattice Λ is not perfectly random because of the presence of unusually short vectors, the next section analyzes how hard it is to detect and find those unusually short vectors.

Remark 6.13. Note that we have $\beta > q$, yet the q-vectors give no proper solution to the SIS instance since it is required that the short solution is non-null modulo q. This is one of the reasons our scheme constraints the ℓ_2 and the ℓ_∞ norms of signature vectors; this ensures that the reduction provides a vector $\mathbf{v}$ such that $\|\mathbf{v}\|_\infty < q/2$, and thus is non-null modulo q. While we could have chosen larger values for B_∞ and still have a valid security reduction, choosing it as small as possible for correctness can only make the scheme more secure.

Quantification. The hardness of this SIS problem is dictated by the ratio $\beta/\sqrt{q}$ and the dimension m, precisely it is necessary to run BKZ with a blocksize providing a Hermite factor $\delta^m < \beta/\sqrt{q}$. The relation between the block-size δ and the running time is interpolated from [CN11].

Margins. The cost given in the last line of Table 6.1 is expressed as the number of nodes to visit in the enumeration tree of the enumeration subroutine of BKZ. Each visit requires about 100 CPU cycles, and BKZ needs to perform at least $2n$ such enumerations, adding an additional 10 bits to those numbers. Yet, those numbers do not directly give rise to an attack as they are derived from a security reduction; actually forging seems to require finding vectors smaller by a factor 2.

Table 6.1 – Hardness of the underlying SIS instance.

Scheme	SIS parameter $\beta/\sqrt{q}$ (as in Theorem 6.8)	Required Block Size	Enumeration Cost $\log_2 T$
BLISS-0	$63 = 1.0083^m$	125	**53**
BLISS-I	$441 = 1.0060^m$	215	**130**
BLISS-II	$409 = 1.0059^m$	220	**136**
BLISS-III	$289 = 1.0055^m$	245	**168**
BLISS-IV	$231 = 1.0053^m$	260	**188**

6.4.3 Primal Lattice Reduction Key Recovery

Attack Overview. The attack consists in applying lattice reduction to the primal lattice Λ hoping that the short vector found will be the secret key. This problem can be seen as a ring variant of the unique-SVP problem.

Cryptanalysis and Experiments on NTRU Lattices. The most complete study of the behavior of BKZ in the presence of unusually short vector(s) is due to Gama and Nguyen [GN08] who thoroughly analyzed the algorithm's running time in the presence of *one* such vector. Their experiments show that the hardness of finding this vector depends on the ratio λ_2/λ_1, that is, the gap between the second-shortest and the shortest vectors in the m-dimensional lattice. In practice, for BKZ-20, the shortest vector was found when $\lambda_2/\lambda_1 > .48 \cdot 1.01^m$.

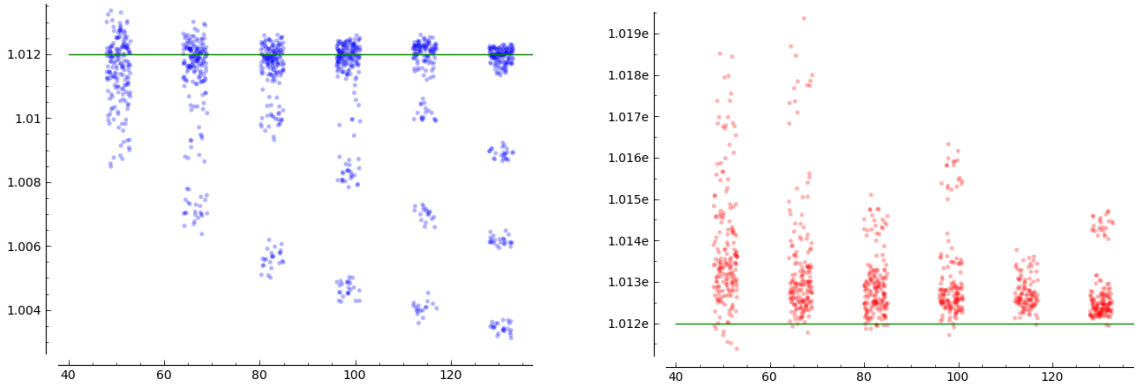
We ran similar experiment of BKZ-20 in the case of $2n$-dimensional NTRU lattices where $\lambda_1 = \ldots = \lambda_n$. In NTRU lattices, the gap normally occurs between the n-th and the $n+1$-th successive minima, and one might think that the ratio between these two quantities would somehow determine the hardness of the instance. Our experiments showed that this is not the case, and the shortest vector was found when $\sqrt{qm/2\pi e}/\lambda_1$ was greater than $.40 \cdot 1.012^m$ (see Figure 6.1). Despite the fact that there is no vector in the lattice having length $\sqrt{qm/2\pi e}$ this is actually consistent with the results of [GN08]! The reason is that $\sqrt{qm/2\pi e}$ is the *expected* length of the shortest vector according to the Gaussian heuristic,[9] and we would also expect $\lambda_2 \approx \sqrt{qm/2\pi e}$ in a random q-ary lattice analyzed in [GN08]. Thus one could say that the hardness of finding a short vector in q-ary lattices depends not on the gap, but rather on the ratio between the Gaussian heuristic and the actual length of the shortest vector.

Similar to the results in [GN08], when the ratio was smaller than $.40 \cdot 1.012^m$, the resulting shortest vector had length about $\sqrt{q} \cdot 1.012^m$. In other words, BKZ-20 behaved as if the lattice were truly random. Because of our experiments with BKZ-20, it seems reasonable to assume that BKZ behaves analogously for larger block sizes. Thus we can measure its efficacy according to the BKZ-2.0 methodology in [CN11]. We would like to stress that we have no explanation for the reason why the ratio between the Gaussian heuristic and the actual length of the vector seems to dictate the hardness of finding short vectors in NTRU lattices. We are equally unsure whether this phenomenon implies that these lattices are indeed as hard as the random lattices that have been more exhaustively studied [GN08, CN11].

6.4.4 Dual Lattice Reduction Key Recovery

Attack Overview. The attack consists in using short dual lattice vectors as a distinguisher for the existence of a very short vector $\mathbf{s}$ in a lattice [MR09]. Then, one may use the distinguisher to completely recover this very short vector using the reduction of Micciancio and Mol [MM11], inspired by the Goldreich-Levin Theorem [GL89].

[9] The Gaussian heuristic says that for certain types of random lattices $\mathcal{L}$, we will have $\lambda_1(\mathcal{L}) \approx \det(\mathcal{L})^{1/m} \cdot \sqrt{\frac{m}{2\pi e}}$ [GN08].



(a) Shortest vector not found

(b) Shortest vector found

Figure 6.1 – Results BKZ-20 for $n \in [48, 150]$, $q \in [6000, 25000]$ and binary search on the λ_1-threshold. On horizontal axis is the value of $n + \texttt{random(0,5)}$ and on vertical axis is $\left(\frac{1}{.40}\sqrt{\frac{qm}{2\pi e}}/\lambda_1\right)^{1/2n}$

Table 6.2 – Cost of finding the Ring-unique shortest vector via primal lattice reduction.

Scheme	Ring-Unique-SVP parameter $\sqrt{\frac{qm}{2\pi e}}/\lambda_1$	Required Block Size	Enumeration Cost $\log_2 T$
BLISS-0	$14 = 1.0051^m$	270	**200**
BLISS-I	$46 = 1.0037^m$	>300	**>240**
BLISS-II	$46 = 1.0037^m$	>300	**>240**
BLISS-III	$30 = 1.0033^m$	>300	**>240**
BLISS-IV	$25 = 1.0031^m$	>300	**>240**

Table 6.3 – Cost of distinguish the existence of the shortest vector via primal lattice reduction.

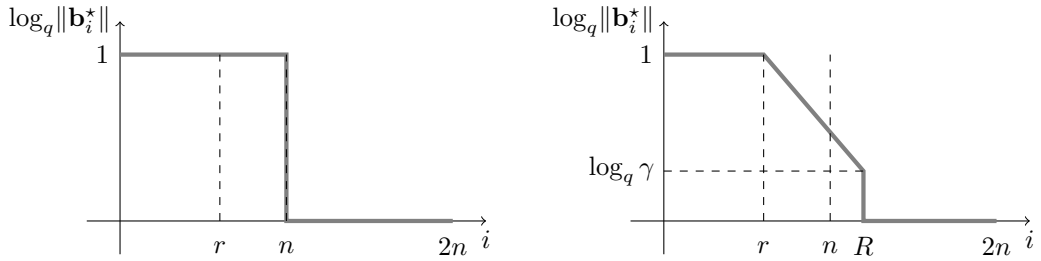
Scheme	Best Block Size b	Enum. Cost $\log_2 T$	Hermite Factor δ	Dist. Adv. $\log_2 \epsilon$	**Total Cost** $\log_2(T/\epsilon)$
BLISS-0	110	45	1.0088^m	-5.5	**56**
BLISS-I	220	136	1.0059^m	-20	**177**
BLISS-II	220	136	1.0059^m	-20	**177**
BLISS-III	240	162	1.0056^m	-19	**201**
BLISS-IV	245	168	1.0056^m	-21	**211**
[Lyu12, **Set-IV**]	190	103	1.0067^m	-7	**118**
[GLP12, **Set-I**]	130	56	1.0081^m	-5	**67**

Quantification. For a q-ary lattice Λ of dimension m, using a vector $\mathbf{v} \in \Lambda^\times$ (where $\Lambda^\times$ is the dual lattice) and assuming its direction is random, one is able to distinguish the existence of an unusual short vector $\mathbf{s}$ in the dual with probability $\epsilon = e^{-\pi\tau^2}$, where $\tau = \|\mathbf{v}\| \cdot \|\mathbf{s}\|/(q\sqrt{m})$.

Next, using this distinguisher as an oracle, it is possible to recover one entry of the private key except with small fixed probability, using $1/\epsilon^2$ calls to that oracle. We then iterated over different block-sizes (5 by 5) to minimize the total cost T/ϵ^2, where T is the running time of the enumeration subroutine of BKZ.

Remark 6.14. Rather than trying to find the proper secret key $\mathbf{s} = (\mathbf{f}, -2\mathbf{g} + 1)^t$ as a short solution to $(2\mathbf{a}_q, 2)\mathbf{s} = \mathbf{0} \bmod q$, one would search directly $\mathbf{s}' = (\mathbf{f}, \mathbf{g})^t$ as a shorter solution to $(\mathbf{a}_q, -1)\mathbf{s}' = 1 \bmod q$.

Margins. To stay on the safe side, we do not include the additional n^2 factor to the running time of this attack: indeed there are n coordinates to guess, and each BKZ reduction requires at least



(a) Before reduction (b) After reduction

Figure 6.2 – Basis Profile during the Hybrid Attack.

n enumerations; one might then be tempted to claim an additional 20 bits of security. Yet it is unclear whether one needs to run the full BKZ reduction to get new short vectors, neither if one can reuse the same short dual vector to guess each coordinate. Even though we do not claim an attack in time 2^{67} on [GLP12], we believe that claiming more than 90 bits of security is a long shot. The difference between our measurement and theirs might be explained by the fact that the authors only considered the case where ϵ was close to 1.

6.4.5 Hybrid MiM-Lattice Key Recovery

Attack Overview. The attack from [HG07] uses lattice reduction as a preprocessing step, in order to decrease the search space of combinatorial attacks. Precisely, one first chooses parameters r and R, applies lattice reduction on the sub-lattice generated by the vectors of the sub-basis $\mathbf{b}_r, \ldots, \mathbf{b}_{R-1}$ (see Figure 6.2), and runs the MiM attack only over the $2n - R$ last coordinates.

In order to perform the combinatorial attack, one needs to obtain a basis whose last orthogonalized vector is large enough. Precisely, the basis needs to be good enough so that Babai's algorithm properly solves BDD on the error $\mathbf{s}' = (s_1, \ldots, s_R, 0, \ldots, 0)$. A necessary condition is therefore:

$$\langle \mathbf{s}', \mathbf{b}_i^\star \rangle / \|\mathbf{b}_i^\star\|^2 \leqslant 1/2 \,, \tag{6.3}$$

where the $\mathbf{b}_1^*, \ldots, \mathbf{b}_R^*$ is the Gram-Schmidt orthogonalization of $\mathbf{b}_1, \ldots, \mathbf{b}_R$.

Quantification. Once again, we assume that the lattice reduction algorithm provides a basis of random direction. Therefore, we model the quantity $\langle \mathbf{s}', \mathbf{b}_i^\star \rangle / \|\mathbf{b}_i^\star\|^2$ as a Gaussian of standard deviation $\|\mathbf{s}'\|/(\sqrt{R}\|\mathbf{b}_i^\star\|)$. Denoting $\gamma = \|\mathbf{b}_{R-1}^\star\|$, one models by the GSA (geometric series assumption) that $\|\mathbf{b}_{R-1-i}^\star\| = \gamma \times \delta^{2i}$, where δ is the Hermite factor. To verify Equation (6.3) with reasonable probability (say at least 0.01), it is required that $\gamma \geqslant 2.5\|\mathbf{s}'\|/\sqrt{R}$.

We thus determine the security against this attack as follows: to claim λ bits of security, set R so that it takes 2^λ time and memory to exhaust the last $2n-R$ entries of the secret. Recall that e denotes the entropy of a single entry, each step of the Meet in the Middle attack requires $O(n^2)$ operations, and at least $e \cdot R$ bits of storage, therefore we set R such that $R \cdot e = 2\lambda - \log_2(e \cdot R) - \log_2(n^2)$.

Then we determine γ, and run BKZ-2.0 simulation according to [CN11], increasing block-size until $\gamma \geqslant 2.5\|\mathbf{s}'\|/\sqrt{R}$. Finally, deduce the cost of lattice reduction and verify it is greater than 2^λ. Note that r is derived from the behavior of this simulation. Analysis results are described in Table 6.4.

Margins. There is a small security margin coming from the fact that we set the parameters so that the attack succeeds with probability 0.01, which would add about 7 bits of security, and again 10 extra bits because BKZ requires at least $2n$ enumeration. More importantly we considered that the attacker has 2^λ memory available; in practice it is unlikely that an attacker may have as much memory available as the number of bit-operations.[10]

[10] In 2007, there were no more than 2^{71} bits of storage globally, while all general-purpose computers could execute 2^{87} operations in a year. Storage growth is 23% a year versus 58% for computing power (see `http:`

Table 6.4 – Hybrid MiM+Lattice Reduction Attack Parameters.

Scheme	**MiM Search Cost** $\log_2 M$	Entropy per sk Entry	MiM Search Dimension R	Required Block Size	**BKZ Enum. Cost**
BLISS-0	**60**	2.11	46	165	**84**
BLISS-I	**128**	1.18	194	245	**168**
BLISS-II	**128**	1.18	194	245	**168**
BLISS-III	**160**	1.60	183	>300	**>200**
BLISS-III	**192**	1.77	201	>300	**>200**
[Lyu12, **Set-IV**]	**100**	1.58	110	220	**150**
[GLP12, **Set-I**]	**80**	1.58	85	140	**60**

6.5 Parameters and Benchmarks

In this section, we first propose parameters sets for the scheme BLISS described in this chapter. Next, we compare the benchmarks of our proof-of-concept implementations with the `openssl` running times of RSA and ECDSA.

6.5.1 Parameters Sets

In Table 6.5, we propose several sets of parameters to implement the scheme described in this chapter. The signature schemes BLISS-I and BLISS-II are respectively optimized for speed and compactness and offer 128 bits of security (*i.e.* long-term protection [NIS11, ECR12]). The signature schemes BLISS-III and BLISS-IV offer respectively 160 and 192 bits of security. The two last lines provide typical security measurement against direct lattice attack in terms of Hermite factor, but slightly better attacks exist. Therefore, our security claims are derived from an extensive analysis based on BKZ-2.0 simulation [CN11] in interaction with other techniques [MR09, MM11, HG07], as detailed in Section 6.4.

One of the objectives of this work was to determine whether the scheme from [Lyu12] could be improved so as it remains sufficiently secure for a dimension $n = 256$. Even though this seems possible when only considering *direct* lattice attacks, it turns out to be slightly out of reach according to the analysis of Section 6.4. Any additional trick might unlock an extremely efficient 80-bit secure signature scheme; it seems to us a challenging but worthwhile goal. We do however propose a toy variant BLISS-0 in this dimension for which we expect up to 60 bits of security. Yet, we believe it would require a significant effort to break this toy variant; we leave it as a challenge to motivate further advance in lattice cryptanalysis. Note that choosing a non power-of-two dimension n would have been possible but yields several unwelcome consequences: on efficiency first as NTT becomes at least twice slower and the geometry is worse (our constant C grows), but also on simplicity as one will no longer work as on the simple quotient by $x^n + 1$. However, it is possible to get about 100 bits of security in dimension $n = 379$ for signatures of size 4kb. In comparison [Lyu12, **Set-IV**] and [GLP12, **Set-I**] have respective signature sizes of 15kb and 9.5kb, for a claimed security of 100 bits.[11]

6.5.2 Timings

In Table 6.6, we provide running times of our proof-of-concept implementation [DL13] of our signature scheme with the parameters provided above, on a desktop computer. We also provide running times for the `openssl` implementations of RSA and ECDSA.[12] Note that, despite the lack of optimization on our proof-of-concept implementation, we derived interesting timings. First, our verification time is nearly the same for each of our variants, and is much faster than the RSA and the (even worse) ECDSA verifications by a factor 10 to 30. Secondly, excluding RSA which is really

`//news.usc.edu/#!/article/29360/How-Much-Information-Is-There-in-the-World`). There are about 2^{160} atoms on earth.

[11] Our analysis in Section 6.4 shows that the security of [GLP12, **Set-I**] may actually be a little lower than claimed.

[12] ECDSA on a prime field $\mathbb{F}_p$: `ecdsap160`, `ecdsap256` and `ecdsap384` in `openssl`.

Table 6.5 – Parameter proposals.

Name of the scheme	**BLISS-0**	**BLISS-I**	**BLISS-II**	**BLISS-III**	**BLISS-IV**
Security	Toy ($\leqslant$ 60 bits)	128 bits	128 bits	160 bits	192 bits
Optimized for	Fun	Speed	Size	Security	Security
n	256	512	512	512	512
Modulus q	7681	12289	12289	12289	12289
Secret key densities δ_1, δ_2	.55 , .15	.3 , 0	.3 , 0	.42 , .03	.45, .06
Gaussian standard deviation σ	100	215	107	250	271
α	.5	1	.5	.7	.55
κ	12	23	23	30	39
Secret key N_κ-Threshold C	1.5	1.62	1.62	1.75	1.88
Dropped bits d in $\mathbf{z}_2$	5	10	10	9	8
Verification thresholds B_2, B_∞	2492, 530	12872, 2100	11074, 1563	10206,1760	9901, 1613
Repetition rate	7.4	1.6	7.4	2.8	5.2
Entropy of challenge $\mathbf{c} \in \mathbb{B}_\kappa^n$	66 bits	132 bits	132 bits	161 bits	195 bits
Signature size	**3.3kb**	**5.6kb**	**5kb**	**6kb**	**6.5kb**
Secret key size	1.5kb	2kb	2kb	3kb	3kb
Public key size	3.3kb	7kb	7kb	7kb	7kb
SIS parameter $\beta/\sqrt{q}$ (as in Theorem 6.8)	$63 = 1.0083^m$	$441 = 1.0060^m$	$409 = 1.0059^m$	$289 = 1.0055^m$	$231 = 1.0053^m$
Ring-Unique-SVP parameter $\sqrt{\frac{qm}{2\pi e}}\Big/\lambda_1$	$14 = 1.0051^m$	$46 = 1.0037^m$	$46 = 1.0037^m$	$30 = 1.0033^m$	$25 = 1.0031^m$

Table 6.6 – Benchmarking on a desktop computer (Intel Core i7 at 3.4Ghz, 32GB RAM) with `openssl` 1.0.1c.

	Security	Signature size	Sign (ms)	Sign/s	Verify (ms)	Verify/s
BLISS-0	$\leqslant$ 60 bits	3.3 kb	0.241	4k	0.017	59k
BLISS-I	128 bits	5.6 kb	0.124	8k	0.030	33k
BLISS-II	128 bits	5 kb	0.480	2k	0.030	33k
BLISS-III	160 bits	6 kb	0.203	5k	0.031	32k
BLISS-IV	192 bits	7 kb	0.375	2.5k	0.032	31k
RSA 1024	72-80 bits	1 kb	0.167	6k	0.004	91k
RSA 2048	103-112 bits	2 kb	1.180	0.8k	0.038	27k
RSA 4096	$\geqslant$ 128 bits	4 kb	8.660	0.1k	0.138	7.5k
ECDSA 160	80 bits	0.32 kb	0.058	17k	0.205	5k
ECDSA 256	128 bits	0.5 kb	0.106	9.5k	0.384	2.5k
ECDSA 384	192 bits	0.75 kb	0.195	5k	0.853	1k

slow, the signature algorithm of BLISS-I is as fast as ECDSA-256 (with the same claimed security). We refer to [NIS11, ECR12] to get the equivalence between the key length of RSA and ECDSA and the expected security in bits.

Besides, we expect our scheme to be much more suitable to embedded devices than both RSA and ECDSA, mainly because our operation are done with a very small modulus (less than 16 bits). By design, the binary representation of q is `11 0000 0000 0001`, that is q has a very small Hamming weight; this structure might yield interesting hardware optimizations. The main issue for such architectures is the generation of discrete Gaussian, addressed in Chapter 4.

6.6 Conclusion

Our contributions in this chapter are five-fold and improve over the lattice-based signature scheme BLISS described in Chapter 5:

1. we propose a new key generation algorithm based on NTRU lattices;
2. we describe several optimizations and implementations tricks (multiplications, efficient hashing, rejection probability computations);
3. we propose a new method to compress the signature;
4. we perform an extensive security analysis to derive concrete parameters;
5. finally, we implement on a 64-bit architecture, and benchmark 5 versions of BLISS (BLISS-0 to BLISS-IV) with different security levels and signature sizes; the proof-of-concept implementations are openly available at [DL13].

As a side result, we show that lattice-based cryptography can be made as efficient (or even more) than classically used algorithms such as RSA or ECDSA, can be adapted to constrained devices and can have reasonable key/signature sizes.

Consequently to our work, implementations of lattice-based signatures schemes began to be described. At SAC 2013, Bansarkhani and Buchmann [BB13] implemented a hash-and-sign signature but their practical results are nowhere near comparable to our parameter sizes and timings. Oder, Pöppelmann and Güneysu [OPG14] implemented our signature scheme on a 32-bit ARM Cortex-M4F RISC and concluded that (1) it is possible to implement a post-quantum, lattice-based signature scheme on a Cortex-M4F microcontroller; (2) our discrete Gaussian sampling algorithm is currently the most efficient (see Chapter 4); and (3) one can obtain an implementation faster than the reference implementations of RSA and ECC for a comparable security level (128 bits).

Also, 15 years after the design of the lattice-based encryption scheme NTRUEncrypt [HPS98], an open-source implementation of the scheme has been made available by the authors at [WEJ13].

The goal of making available this reference code is to enable "more widespread adoption of this superior cryptographic technology".

Lattice-based signatures are still being investigated (*e.g.* [HPS+13, LLLS13, BG14, LLNW14]), and we believe significant improvements could be obtained in the following couple of years.

Note however that the general dearth of lattice cryptanalysis papers stands in contrast to the vast number of articles proposing theoretical lattice-based constructions. Our belief is that this lack of cryptanalytic effort is in part due to the fact that most of the papers with scheme proposals give no concrete targets to attack. One of the proposed instantiations in the present work is a "toy example" (optimized for "fun") that we estimate to have approximately 60 bits of security. Thus if it turns out that NTRU lattices are weaker than believed, it is wholly possible that this example could be broken on a personal computer, and we think this would be of great interest to the practical community. In addition, it could be argued that we do not yet know enough about lattice reduction to be able to propose such "fine-grained" security estimates like 160-bit or 192-bit. One of our hope is that providing practical parameters will spur on the cryptanalysis of our scheme (and therefore of lattice-based cryptography), which is much needed.

Finally, as already suggested in Chapter 4 about Gaussian sampling, a fundamental open problem is to evaluate the resistance of implementations of lattice-based cryptography against side-channel attacks, and designing adapted countermeasures. Also it might be worth investigating software-hardware co-design for lattice-based cryptography, to further enhance its performance and make its deployment possible as soon as some new standards will have emerged.

Part Two

Helping Fully Homomorphic Encryption Become Practical

Overview

In this age of social networks and sharing, it is challenging to provide users both privacy (via encryption) and usability. We believe that encryption should be nearly *transparent* to the user in order to be widely adopted, as illustrated by the poor adoption of PGP [WT99, BHL13].[1] Note that PGP provides cryptographic privacy and authentication for *data communication*. In a cloud setting however, a user might want to ensure the privacy of her *own* data while using the usability benefits from the cloud. In such a context, PGP would only allow to store and retrieve encrypted data. The notion of privacy homomorphism was introduced in 1978 by Rivest, Adleman and Dertouzos [RAD78] to go beyond the storage and retrieval of encrypted data by permitting encrypted data to be *operated on* for interesting operations, in a public fashion – *i.e.* without decrypting the ciphertexts. A typical example is that of a user sending its own encrypted content to the cloud for treatment. Note that the encryption program can be included *by design* in the program, without relying on whether or not other users use this mechanism.

Fully homomorphic encryption (FHE) allows a worker to perform implicit additions and multiplications on plaintext values while exclusively manipulating encrypted data. In other words, it is an encryption scheme that allows, from ciphertexts $E(a)$ and $E(b)$ encrypting bits a, b, to obtain encryptions of $\neg a$, $a \wedge b$ and $a \vee b$ without using the secret key. Clearly, this allows to publicly evaluate any Boolean circuit given encryptions of the input bits. The first construction of a fully homomorphic scheme (based on ideal lattices) was described by Gentry in [Gen09], and his breakthrough idea was to homomorphically evaluate the decryption circuit of a scheme that allowed only shallow circuits to be evaluated to "refresh" ciphertexts. By repeatedly refreshing ciphertexts, the number of homomorphic operations becomes unlimited, resulting in a fully homomorphic encryption scheme.

Unfortunately the huge gain of functionalities came to the cost of practicality – the first referenced implementation encrypting bits was proposed by Gentry and Halevi in [GH11b], uses a public key of 2.25 GB and after *each* multiplication of two encrypted bits, a refresh procedure taking 31 minutes, on a 64-bit Intel Xeon E5450 processor running at 3GHz, was *required*. This active research area yielded new schemes [vDGHV10, BV11a, BGV12, LTV12, GSW13] based on different assumptions, many improvements [SV10, OYKU10, SS10, GH11b, SV11, LMSV11, CMNT11, BV11b, GH11a, SS11a, GHS12a, CNT12, GHS12b, Bra12, FV12, BGH13, GHPS13, CCK$^+$13, ASP13, BV14, BLLN13, CLT14a, LN14a] and implementation results [GH11b, CMNT11, NLV11, PBS11a, CNT12, GHS12c, FSF$^+$13, MHM$^+$13, BLLN13, CLT14a, LN14a]. Note however that, comparatively, very few implementations of homomorphic schemes are publicly available [PBS11b, CT12, HS13, Lep14].

This part of the thesis presents several contributions towards the practicality of fully homomorphic encryption schemes [CCK$^+$13, LP13, CLT14b].

[1] For PGP to be of any use, both end-users need to have a working PGP program. Even though developers proposed and are still proposing graphical user interfaces to PGP, adopting it remains out of reach of most users. And unfortunately, the current poor adoption of this encryption program does not encourage anybody to use it.

First, we propose two extensions of the fully homomorphic encryption scheme over the integers of van Dijk *et al.* (DGHV) [vDGHV10]. In Chapter 7, we describe a batch variant of the DGHV scheme, *i.e.* a variant of DGHV that supports encrypting and homomorphically processing a vector of plaintext bits as a single ciphertext instead of a single bit. In Chapter 8, we describe a leveled variant of the DGHV scheme with the scale-invariant property [Bra12]. The resulting scheme has a single secret modulus whose size is linear in the multiplicative depth of the circuit to be homomorphically evaluated, instead of exponential. In a nutshell, these improvements transformed the initial impractical DGHV scheme into a serious candidate for an efficient fully homomorphic encryption primitive.

Next, we tackle the bottleneck in FHE performances, namely the bootstrapping procedure, by proposing a method to compute the *exact* minimal number of bootstrappings required to homomorphically evaluate *any* circuit. We successfully applied our method to a range of real-world circuits that perform various operations over plaintext bits. Practical results show that some of these circuits benefit from significant improvements over the naive evaluation method where all multiplication outputs are bootstrapped.

Finally, we illustrate the practicality of our new schemes with proof-of-concept implementations in C++, and by homomorphically evaluating the AES block cipher [FIP01]. We obtain timings comparable to the timings presented by Gentry, Halevi and Smart at Crypto 2012 for their implementation of a Ring-LWE based fully homomorphic encryption scheme [GHS12c].

Chapter 7

Batch Fully Homomorphic Encryption over the Integers

7.1 Introduction

In this chapter, we first recall and discuss attacks against the Approximate-GCD problem, and introduce variants thereof. These variants allow us to revisit the fully homomorphic encryption scheme over the integers of van Dijk, Gentry, Halevi and Vaikuntanathan (DGHV) [vDGHV10], and to extend it to batch fully homomorphic encryption, *i.e.* to a scheme that supports encrypting and homomorphically processing a vector of plaintext elements as a single ciphertext. Our variant is semantically secure under the decisional Error-Free Approximate-GCD problem, which we prove to be equivalent to the computational Error-Free Approximate-GCD problem considered in [vDGHV10]. We also show how to perform arbitrary permutations on the underlying plaintext vector given the ciphertext and the public key. Finally, we explain how to derive concrete parameters and benchmark our scheme on a desktop computer.

Part of this chapter consists of the article Batch Fully Homomorphic Encryption over the Integers [CCK+13], cosigned with J.H. Cheon, J.-S. Coron, J. Kim, M.S. Lee, T. Lepoint, M. Tibouchi and A. Yun, and published at Eurocrypt 2013 [JN13]. This latter article is a merged version of a work [CLT13a] cosigned with J.-S. Coron and M. Tibouchi, and the work [KLYC13] of J. Kim, M.S. Lee, A. Yun and J.H. Cheon. Section 7.2.2 of this chapter is part of the article Scale-Invariant Fully Homomorphic Encryption over the Integers [CLT14a], cosigned with J.-S. Coron and M. Tibouchi, and published at PKC 2014 [Kra14]. Finally, part of Section 7.2 was done while the author was an intern in the Cryptography Research group at Microsoft Research in Redmond, under the supervision of Kristin Lauter and Michael Naehrig.

7.1.1 Background on Fully-Homomorphic Encryption

Homomorphic encryption is a special kind of encryption that allows to operate publicly on ciphertexts without knowing the decryption key. For example, an additively homomorphic encryption scheme allows to combine a ciphertext c of m and a ciphertext c' of m' to obtain a ciphertext of $m + m'$. This homomorphic feature enables one to design rich and complex protocols with surprising and exciting applications. An oversimplified example consists in using the additive property of an additively homomorphic encryption scheme to instantiate an encrypted electronic voting scheme. The problem of designing a *fully* homomorphic encryption scheme, that is a scheme that allows evaluation of arbitrarily complex programs on encrypted data, was suggested by Rivest, Adleman and Dertouzos [RAD78] back in 1978. Partial progress has been achieved during thirty years [GM82, Pai99, BGN05, IP07], but fully homomorphic encryption remained a "never-to-be-found Holy Grail of cryptography" [Mic10] [sic] until the breakthrough result of Gentry in 2009 [Gen09].

Gentry proposes the first construction of a fully homomorphic scheme (based on ideal lattices), and proceeds as follows. First, one constructs a *somewhat homomorphic* encryption scheme, that is an encryption scheme that only supports a limited number of homomorphic operations (especially of multiplications). A ciphertext contains some noise that becomes larger with successive homomorphic

operations, and only ciphertexts whose noise size remains below a certain threshold can be decrypted correctly. The second step is to *squash* the decryption procedure associated with an arbitrary ciphertext so that it can be expressed as a low degree polynomial in the secret key bits. Then, Gentry's key idea, called *bootstrapping*, consists in homomorphically evaluating this decryption polynomial on encryptions of the secret key bits, resulting in a different ciphertext associated with the same plaintext, but with possibly reduced noise.[1] This *refreshed* ciphertext can then be used in subsequent homomorphic operations; by repeating this operation, the number of homomorphic operations becomes unlimited, resulting in a fully homomorphic encryption scheme.

Unfortunately, even though the somewhat homomorphic encryption scheme was quite practical for shallow circuits [SV10], the full scheme practicality remained an important open problem. In 2011, an implementation of the full Gentry's scheme was proposed by Gentry and Halevi in [GH11b] with a public key of 2.3 GB and a ciphertext refresh procedure of 30 minutes. Therefore, to homomorphically evaluate an AND gate in a given circuit, one has to spend 30 minutes to refresh the ciphertext in order to continue the computation... The implementation is nonetheless based on many interesting algorithmic optimizations, including some borrowed from Smart and Vercauteren [SV10] and from Stehlé and Steinfield [SS10]. Some improvements to Gentry's key-generation procedure are discussed in [OYKU10, SS11a].

Gentry's breakthrough made possible to design fully homomorphic encryption (FHE) schemes based on different hardness assumptions.

Among them, the possibly conceptually simplest FHE scheme was proposed by van Dijk, Gentry, Halevi and Vaikuntanathan [vDGHV10]. Starting from an arguably "simplest" somewhat homomorphic encryption scheme, they use Gentry's blueprint to produce a fully homomorphic encryption scheme. The resulting scheme demonstrates that FHE can be achieved only by elementary means, and does not require the complexity of ideal lattices. The scheme is proved to be secure under the Approximate-GCD problem, that is the problem of computing a greatest common divisor from approximations of multiples of this integer. Unfortunately, to resist known attacks against the Approximate-GCD problem, the parameters constraints yield a public key size of $\mathcal{O}(\lambda^{10})$ where λ is the security parameter, which is too large for any practical system. Once again the primary open problem was to improve the efficiency of this conceptually simple scheme while preserving the hardness of the Approximate-GCD problem. A first step towards the practicality of this scheme was described by Coron, Mandal, Naccache and Tibouchi in [CMNT11]. The authors reduce the public key size to $\mathcal{O}(\lambda^7)$ by storing only a subset of it and generating it on the fly by computing the elements multiplicatively, at the cost of working with an exact multiple of the secret key (already suggested in [vDGHV10]); that is in the *error-free setting*. Next the authors describe the first implementation of the resulting scheme, incorporating among others some of the optimizations of [SV10, SS10], and obtained similar performances as the Gentry-Halevi implementation, with a public key of 800MB and a refresh procedure of 14 minutes. Another step towards a practical scheme was achieved by Coron, Naccache and Tibouchi in [CNT12] thanks to a compression technique that reduces the public key size by several orders of magnitude. They then describe an implementation of this new variant, still in the error-free setting, with performances similar to [CMNT11] but a 10.3MB public key. A public implementation in SAGE [S$^+$14] is available at [CT12].

Armed of these two schemes and confident on the fact that FHE is achievable by simple means (thanks to [vDGHV10]), a third very interesting scheme was proposed by Brakerski and Vaikuntanathan in [BV11a]. Contrary to [Gen09, vDGHV10], this scheme deviates from the squashing paradigm and is based *solely*[2] on the (standard) learning with errors (LWE) assumption – and therefore on the (well-known, although still not well-understood) worst-case hardness of "short vector problems" on arbitrary lattices (see Section 3.2). This FHE scheme has known

[1] Indeed, a natural idea is that to "refresh" a ciphertext whose noise is almost too big to obtain a new ciphertext with a shorter noise, it suffices to decrypt it and re-encrypt it. And that is the idea behind bootstrapping: we do allow decryption, but homomorphically!

[2] Additionally to the performance bottleneck of Gentry's scheme and the scheme of [vDGHV10], the latter schemes have to rely on a relatively strong computational assumption. Namely, the main caveat of these schemes is that the squashing step forced the authors to rely on the hardness of the (average-case) sparse subset-sum problem. Brakerski and Vaikuntanathan get rid of the squashing step by a *dimension-modulus reduction* technique.

numerous modifications and improvements, notably a variant over ring [BV11b], exponential improvements in the noise growth [BGV12], NTRU-based schemes [LTV12, BLLN13], scale-invariant schemes [Bra12, FV12]. A well known implementation of [BGV12] is described in [GHS12c] (and openly available at [HS13]), in which a full-fledge AES circuit is homomorphically evaluated in several dozens of hours.

At Crypto 2013, Gentry, Sahai and Waters [GSW13] describe a "conceptually simpler [and] asymptotically faster" FHE scheme based on LWE, that does not use the "relinearization" that is at the heart of previously mentioned LWE-base FHE schemes. Indeed, this "relinearization" step did not appear to be very natural, is expensive and slightly intricate. In [GSW13], a conceptually simpler scheme with a "natural" multiplication procedure (over matrices) is proposed. Unfortunately, and as rightly explained by the authors in their introduction, the performance of the scheme is worse than the best-known LWE-based schemes. The scheme is sold as an asymptotically efficient scheme that presents new techniques. Interestingly enough, this scheme has led the first FHE scheme as secure as public-key encryption schemes by Brakerski and Vaikuntanathan [BV14], that is a scheme whose hardness matches the best known hardness for "regular" lattice-based public-key encryption scheme (instead of the quasipolynomial approximation achieved in the other LWE proposals). Exciting results are starting to be built on these schemes [ASP14], and improving its efficiency could unlock a very promising FHE scheme.

Finally, note that a number of implementation benchmarks have been reported in the literature [GH11b, CMNT11, NLV11, PBS11a, CNT12, GHS12c, FSF+13, MHM+13, BLLN13]. Unfortunately, making implementations available is not standard behavior in the cryptographic community. Therefore, even though fully homomorphic encryption becomes more and more efficient, very few implementations of homomorphic schemes are publicly available [PBS11b, CT12, HS13, Lep14].

In the rest of this introduction, we would like to emphasize two topics about homomorphic encryption that will be basic notions on which our results will develop in this part.

Batching. In a series of works [SV10, SV11], Smart and Vercauteren observed that using the Chinese Remainder Theorem in number fields allows to encrypt a vector of "plaintext slots" (instead of a single element), and that a single homomorphic operation implicitly performs the componentwise operation over the plaintext vector. They used this observation for *batch* (or SIMD [HP07]) homomorphic operations. In other words, a function f can be homomorphically evaluated ℓ times in parallel on ℓ different inputs with approximately the same cost than a single evaluation of f on a single input. This technique was adapted to the LWE-based schemes in [GHS12b, BGH13].

Batching in homomorphic cryptography is really useful when the same operation has to be applied on each slot. But it could also be used as a way to reduce the overhead complexity of homomorphic computation in general, *i.e.* the ratio of encrypted computation complexity to unencrypted computation complexity. To reduce this overhead, one needs a method of moving data between slots in each SIMD word, as in normal program execution of SIMD instructions (*e.g.* the SSE instructions on x86) [GHS12b].

Modulus-Switching and Scale-Invariance. The modulus-switching technique has been introduced by [BGV12] (as a refinement of [BV11a, BV11b]). Using this technique, the noise ceiling increases only linearly with the multiplicative depth, instead of increasing exponentially. They introduce the notion of *leveled* FHE scheme as a scheme whose parameters depend (polynomially) on the depth of the circuits that the scheme is capable of evaluating. The essence of the modulus-switching technique is that a ciphertext $c \bmod q$ can be publicly transformed into a valid ciphertext $c \bmod p$ while preserving correctness, by a simple scaling by (p/q) and appropriate rounding. This technique was successfully adapted on the scheme over the integers [vDGHV10] by Coron, Naccache and Tibouchi in [CNT12]. This technique introduces a ladder of L moduli to evaluate a circuit of multiplicative depth L, which impacts the size of the public key (*e.g.* [GHS12c] need to run on a server with 256GB of memory).

In [Bra12], Brakerski introduces a new technique called *scale-invariance* that allows to use the same modulus throughout the evaluation process, reducing the size of the public key; the resulting

scheme is still a leveled FHE scheme, that is the noise growth is still linear in the depth of the circuit evaluated. This technique was adapted to the RLWE-based scheme [BV11b] by Fan and Vercauteren in [FV12], and to the NTRU-based scheme [LTV12] by Bos, Lauter, Loftus and Naehrig in [BLLN13].

In this part, we focus on the fully homomorphic encryption scheme over the integers DGHV (van Dijk, Gentry, Halevi and Vaikuntanathan [vDGHV10]). Namely, we will improve further upon this scheme by simplifying it and building a variant that encrypts bit-vectors instead of single bits (Chapters 7 and 8), *i.e.* we batch the DGHV scheme. In Chapter 8, we show how to adapt the scale-invariance technique to the DGHV scheme, thus yielding an efficient leveled FHE scheme over the integers. Our variants are shown to be secure under the same Approximate-GCD problem initially considered while providing performance enhancements that will eventually allow us to homomorphically evaluate a full-fledged AES circuit (Chapter 10).

We recall the initial DGHV scheme in Section 7.1.2, and explain our contributions and techniques to batch the DGHV scheme in Section 7.1.3.

Notation. Throughout the chapter, λ denotes the security parameter and $[\cdot]_q$ denotes the reduction modulo q into the interval $(-q/2, q/2]$ of an integer.

7.1.2 The Somewhat Homomorphic DGHV Scheme

A conceptually simple FHE scheme (DGHV) was proposed in [vDGHV10], and was subsequently improved by Coron and others [CMNT11, CNT12]. In this section, we briefly recall the somewhat homomorphic encryption scheme described in [vDGHV10] (which encrypts bits); we refer to Section 7.3.1 for details on security and parameter constraints, and to Section 7.4 for transforming it into a *fully* homomorphic encryption scheme (that can therefore homomorphically evaluate any – polynomially bounded – boolean circuit).

For a specific η-bit odd integer p, we use the following distribution over γ-bit integers:

$$D_{\gamma,\rho}(p) = \{q \cdot p + r : q \leftarrow [0, 2^\gamma/p), r \leftarrow (-2^\rho, 2^\rho)\}\,.$$

DGHV.Keygen(1^λ). Generate an η-bit random prime integer p. For $0 \leqslant i \leqslant \tau$, sample $x_i \leftarrow D_{\gamma,\rho}(p)$. Relabel the x_i's so that x_0 is the largest. Restart unless x_0 is odd and $[x_0]_p$ is even. Let $\mathsf{pk} = (x_0, x_1, \ldots x_\tau)$ and $\mathsf{sk} = p$.

DGHV.Encrypt($\mathsf{pk}, m \in \{0,1\}$). Choose a random subset $S \subseteq \{1, 2, \ldots, \tau\}$ and a random integer r in $(-2^{\rho'}, 2^{\rho'})$, and output the ciphertext:

$$c = \left[m + 2 \cdot r + 2 \cdot \sum_{i \in S} x_i \right]_{x_0}. \tag{7.1}$$

DGHV.Eval($\mathsf{pk}, C, c_1, \ldots, c_t$). Given the boolean circuit C with t input bits and t ciphertexts c_i, apply the addition and multiplication gates of C to the ciphertexts, performing all the additions and multiplications over the integers, and return the resulting integer.

DGHV.Decrypt(sk, c). Output $m \leftarrow [c \bmod p]_2$.

As shown in [vDGHV10] the scheme is somewhat homomorphic, *i.e.* a limited number of homomorphic operations can be performed on ciphertexts. More precisely given two ciphertexts $c = q \cdot p + 2r + m$ and $c' = q' \cdot p + 2r' + m'$ where r and r' are ρ'-bit integers, the ciphertext $c + c'$ is an encryption of $m + m' \bmod 2$ under a $(\rho' + 1)$-bit noise and the ciphertext $c \cdot c'$ is an encryption of $m \cdot m'$ with noise bit-length $\simeq 2\rho'$. Since the ciphertext noise must remain smaller than p to maintain correctness, the scheme roughly allows η/ρ' successive multiplications on ciphertexts.

The scheme is semantically secure under the computational Approximate-GCD assumption (cf. Section 7.2).

Remark 7.1. Note that during encryption, a noise $(2 \cdot r)$ of $(\rho' + 1)$ bits is added to the ciphertext. This additional term is not a correctness requirement but a security requirement. Indeed, it is used in [vDGHV10] to statistically drown the contributions of the noises r_i from the x_i of the subset sum in a fresh ciphertext; for the proof one would like to apply the Leftover Hash Lemma over these latter noises, but they sum over $\mathbb{Z}$ and not modulo an integer as in Section 3.3.1. Unfortunately, if the r_i are ρ-bit integers, this means that $\rho' \approx \rho + \lambda \approx 2\lambda$, which significantly increases the somewhat homomorphic encryption scheme's parameters.

In Section 7.3.1, we describe a variant of this scheme without this additional noise (and therefore with smaller parameters) secure under a decisional variant of the Approximate-GCD assumption. This decisional assumption (with noise ρ) is proved to be equivalent to the computational assumption (with noise $\rho' \approx \rho + \lambda$), when an exact multiple of p is available, in Section 7.2.2.

7.1.3 Our Contributions and Techniques

Our main goal is to extend the DGHV scheme of above to support the same *batching* capability [SV10, SV11] as in LWE-based schemes [GHS12b, BGH13], in order to obtain a FHE scheme with similar features but based on different techniques and assumptions.

In the original DGHV scheme, a ciphertext of a bit message $m \in \{0, 1\}$ has the form

$$c = q \cdot p + 2r + m$$

where p is the secret key, q is a large random integer, and r is a small random integer (noise). To encrypt multiple bits m_i into a single ciphertext c, we use the Chinese Remainder Theorem with respect to a tuple of ℓ coprime integers $p_1, \ldots, p_\ell$, of product π. The batch ciphertext has the form

$$c = q \cdot \pi + \mathsf{CRT}_{p_1,\ldots,p_\ell}(2r_1 + m_1, \ldots, 2r_\ell + m_\ell),$$

and correctly decrypts to the bit vector $(m_i)_i$, given by $m_i = [c \bmod p_i]_2$ for all $1 \leqslant i \leqslant \ell$. Modulo each of the p_i's the ciphertext c behaves as in the original DGHV scheme. Accordingly, the addition or multiplication of two ciphertexts yields a new ciphertext that decrypts to the componentwise sum or product modulo 2 of the original plaintexts.

The main challenge, however, is to prove the semantic security of the resulting scheme. In the original DGHV scheme, public-key encryption is performed by masking the message m with a random subset sum of the public key elements $x_j = q_j \cdot p + r_j$ as

$$c = \left[m + 2 \cdot r + 2 \sum_{j=1}^{\tau} b_j \cdot x_j \right]_{x_0}. \tag{7.2}$$

The semantic security is proved in [vDGHV10] by applying the Leftover Hash Lemma (*cf.* Section 3.3.1) on the subset sum, and using the random $2 \cdot r$ in (7.2) to further randomize the ciphertext modulo p (see also Remark 7.1).

Extending DGHV public-key encryption to the batch setting may at first seem straightforward: one can use a similar random subset sum technique in the batch variant by generating public key elements x_j with a small residue modulo each of the p_i's instead of only modulo p. However, for the proof of semantic security to go through, the ciphertext c should then be independently randomized modulo each of the p_i's, which is not easy to achieve without knowing the p_i's. Indeed, if we only use a single additive term $2r$ as in Equation (7.2), then the *same* random term $2 \cdot r = 2 \cdot r \bmod p_i$ is added modulo each of the p_i, which breaks the security proof.

Our main contribution in this chapter is to provide a provably semantically secure generalization of DGHV to the batch setting, *i.e.* over multiple plaintext slots. For this purpose, we introduce a *decisional* variant of the Approximate-GCD problem that we prove to be equivalent to the computation Approximate-GCD problem, when an exact multiple x_0 of p is available, in Section 7.2.2. Under this assumption the integers x_j in the subset-sum from Equation (7.2) are assumed to be indistinguishable from random modulo x_0; semantic security is then proved by applying the Leftover Hash Lemma modulo x_0. As a consequence, the additional random $2 \cdot r$ in Equation (7.2) becomes

unnecessary. Extending DGHV public-key encryption to the batch setting is then straightforward; namely one can use the same random subset sum technique with public key elements x_j having a small residue modulo each of the p_i's instead of only modulo p. We show that our multi-slot DGHV scheme can encrypt $\ell = \tilde{\mathcal{O}}(\lambda^2)$ bits in a single ciphertext; therefore the ciphertext expansion ratio becomes $\tilde{\mathcal{O}}(\lambda^3)$ instead of $\tilde{\mathcal{O}}(\lambda^5)$ in the original scheme.

In addition to componentwise addition and multiplication, we also show how to perform any permutation on plaintext bits publicly. As opposed to [BGV12, GHS12b], we cannot use an underlying algebraic structure to perform rotations over plaintext bits (clearly, the automorphisms of $\mathbb{Z}$ do not provide any useful action on ciphertexts). Instead we show how to perform arbitrary permutations on the plaintext vector during the ciphertext refresh operation at no additional cost (but with a slight increase of the public key size). Our Recrypt operation is done in parallel over the ℓ slots, with the same complexity as a single Recrypt operation in the original scheme.

Finally, we describe an implementation of our multi-slot DGHV scheme, with concrete parameters. While our batch variant of DGHV does not provide additional features nor significantly improved efficiency over the RLWE-based scheme of [GHS12b], we believe it is interesting to obtain FHE schemes with similar properties but based on different techniques and assumptions.

Outline of the Chapter. In Section 7.2, we recall the computational Approximate-GCD problem, and its error-free variant. We also introduce a new decisional error-free variant, that we prove to be equivalent to the computational error-free variant. Finally we review the existing attacks against the Approximate-GCD problems and explain how to derive parameters for a target level of security.

Next in Section 7.3, we simplify the initial somewhat-homomorphic (one-slot) DGHV scheme and extend it to a multi-slot DGHV scheme. The semantic security of the latter schemes is proven under the decisional Error-Free Approximate-GCD assumption.

In Section 7.4, we explain how to make the schemes of Section 7.3 *fully* homomorphic using Gentry's blueprint: first squash the decryption circuit and then apply the bootstrapping technique, *i.e.* homomorphically evaluate the circuit. A complete description of our schemes with a compressed public key is then provided in Section 7.5.

Finally, in Section 7.6 we provide implementation benchmarks and we conclude in Section 7.7.

7.2 The Approximate-GCD Problems

The *approximate greatest common divisor* (Approximate-GCD) problem was introduced in 2001 by Howgrave-Graham [HG01]. Roughly speaking, the problem is to compute the greatest common divisor (GCD) of two numbers x and y, given only approximations $\hat{x}$ and $\hat{y}$ thereof. In [vDGHV10], a generalized variant of the problem, given many approximations instead of 2, is introduced and becomes the hardness assumption of the conceptually simple SWHE scheme described in Section 7.1.2.

The Approximate-GCD problem and some variants have been used in follow-up works by Coron and others [vDGHV10, CMNT11, CNT12, CCK+13, CLT14a]. In this section, we homogenize these definitions and variants; in particular we focus on the Error-Free Approximate-GCD problem that is the building block of the most efficient integer-based FHE schemes.

The Approximate-GCD problem (AGCD) is parametrized by integers $\gamma, \eta, \rho \in \mathbb{N}$.[3]

Definition 7.2. Let $\gamma, \eta, \rho \in \mathbb{N}$. The *AGCD distribution* $D_\rho(p, q_0)$ for a given η-bit integer p and a uniformly chosen $q_0 \in [0, 2^\gamma/p)$ is the set of integers $x_i = q_i \cdot p + r_i$ where $q_i \in [0, q_0)$ and $r_i \in [0, 2^\rho)$ are sampled uniformly (note that when $2^\rho < p$, we have $x_i < q_0 \cdot p$).

[3] In the initial versions of the Approximate-GCD problem [HG01, vDGHV10], the q_i's were uniformly sampled from $[0, 2^\gamma/p)$ instead of $[0, q_0)$ in the recent key generation algorithms [CMNT11, CCK+13] (see also Section 7.1.2). Now the security proofs of the semantic security of the FHE schemes use [vDGHV10, Lemma 4.4] which assumes that the q_i's are uniformly distributed modulo q_0 (this is not the case when they are sampled from the former distribution) and the error-free variant that we present below often makes the latter choice of interval. Therefore we focus on this latter interval in our homogenized definitions. Note also that we only consider integers $r \in [0, 2^\rho)$ instead of $(-2^\rho, 2^\rho)$ in [vDGHV10, CMNT11, CNT12, CN12, CCK+13]. One can always go from one distribution to another by an appropriate centering.

The *computational-AGCD* problem is: For a η-bit integer p and a uniformly chosen $q_0 \in [0, 2^\gamma/p)$, given polynomially many samples from $D_\rho(p, q_0)$, to compute p.

To assess the hardness of this problem, Howgrave-Graham considers two kinds of attacks: a continued fraction approach and a lattice-based approach [HG01]. The latter attack consists in finding small roots of a multivariate polynomial, based on the works on Coppersmith, Boneh, Durfee and Howgrave-Graham [Cop97, BDHG99], and is later generalized by Cohn and Heninger in [CH12]. Both these attacks are adapted to the variant with many approximations in [vDGHV10]. They also describe a brute-force attack on the noise, and two other lattice-based attacks to derive parameters; later extended in [CMNT11, CNT12, CN12]. We will discuss these attacks in the Error-Free settings in Section 7.2.4.

Remark 7.3. In [CMNT11, CNT12, CCK+13], for simplicity of presentation, p is assumed to be prime. Note that the security proofs hold without this condition.

7.2.1 Error-Free Variants of the Computational Approximate-GCD problem

A variant of the Approximate-GCD problem consists in working with *one* exact multiple of p (without noise); more precisely, it assumes that the number $x_0 = q_0 \cdot p$ is also known to the adversary. Even though this variant seemed clearly easier than the classical Approximate-GCD problem, this was only demonstrated in 2012 by Chen and Nguyen [CN12], but the complexity of the algorithm remains exponential in the noise size ρ. It is worth noting however that there is obviously a trivial *quantum* attack against this variant (namely, to factorize x_0).

Revealing an exact multiple of p has important implications for the FHE schemes. However, with the notable exception of [CMNT11], no condition on q_0 was explicitly specified in the problem definition in [vDGHV10, CNT12] (even though it was correctly specified during the key generation). Now, assume a factor $f > 1$ of x_0 can be efficiently recovered; defining $x_0' = x_0/f$, it holds that all the samples x_i can be reduced mod $x_0' < x_0$ and this leads to a smaller Approximate-GCD instance. Therefore one should make sure that it is hard to recover any factor $f > 1$ of x_0. Efficient known methods to factor x_0 are the General Number Field Sieve (GNFS) [LJMP90] and the Elliptic Curve Method (ECM) [Len87].

GNFS: The complexity of this method depends on the size γ of the element x_0 being factorized. We use the last complexity estimate provided in [ECR12] for an RSA modulus, *i.e.* we assume that factoring a γ-bit integer takes at least $2^{\lambda_{\mathsf{GNFS}}}$ elementary operations where:

$$\lambda_{\mathsf{GNFS}} = \Big(\frac{64}{9}\Big)^{1/3} (\gamma \ln 2)^{1/3} \big(\ln(\gamma \ln 2)\big)^{2/3} - 14\,.$$

ECM: This method has complexity sub-exponential in the size of the smallest factor. Assume that x_0 is 2^ν-rough.[4] We use the estimates of [CGPV10] (based on the survey [ZD06]), *i.e.* we assume that factoring a 2^ν-rough integer with ECM requires at least $2^{\lambda_{\mathsf{ECM}}}$ elementary operations, where:

$$\lambda_{\mathsf{ECM}} = \frac{1}{\ln 2} \cdot (2\nu \ln(\nu \ln 2))^{1/2} + 5\,.$$

We provide some lower bounds on γ and ν for usual security levels $\lambda = 80, 100$ and 128 in Table 7.1; note that for the FHE parameters in this chapter, the parameters will be selected so that factoring x_0 is intractable.

Now we generalize the definitions of the Error-Free Approximate-GCD problem mentioned in [vDGHV10, CMNT11, CNT12, CN12, CCK+13] and parametrize it by integers $\gamma, \eta, \nu, \rho \in \mathbb{N}$.

Definition 7.4. Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$. The *computational Error-Free-AGCD* (EF-AGCD) problem is: For a 2^ν-rough η-bit integer p and a uniformly chosen 2^ν-rough $q_0 \in [0, 2^\gamma/p)$, given $x_0 = q_0 \cdot p$ and polynomially many samples from $D_\rho(p, q_0)$, to compute p. The *decisional Error-Free-AGCD* problem is: For a 2^ν-rough η-bit integer p and a uniformly chosen 2^ν-rough $q_0 \in [0, 2^\gamma/p)$, given $x_0 = q_0 \cdot p$ and polynomially many samples from $\mathbb{Z}_{x_0}$ to distinguish whether the samples are distributed uniformly or whether they are distributed according to the AGCD distribution $D_\rho(p, q_0)$.

[4] An integer is said to be a-rough when it does not contain prime factors smaller than a.

Table 7.1 – Some lower bounds on γ and ν for usual security levels λ.

Security level λ	80	100	128
Lower bound on γ	2911	4615	7851
Lower bound on ν	261	388	603

Algorithm 7.1 Learn-LSB.

function Learn-LSB($z = qp + r \in [0, 2^\gamma)$ with $|r| \leqslant 2^\rho$, $x_0 = q_0 \cdot p$)
  Generate $x_1, \ldots, x_\tau \leftarrow D_\rho(p, q_0)$
  for $j = 1$ to $\mathsf{poly}(\lambda/\epsilon)$ **do**
    Choose randomly and uniformly a noise $r_j \leftarrow [0, 2^{\rho'})$, a bit $\delta \leftarrow \{0, 1\}$ and a random subset $S_j \subset \{1, \ldots, \tau\}$
    Set $y_j = z + \delta + 2r_j + 2\sum_{i \in S_j} x_i \bmod x_0$
    Call $\mathcal{A}$ to get a prediction of $(r \bmod 2) \oplus \delta$: $a_j \leftarrow \mathcal{A}(y_j)$
    Set $b_j \leftarrow a_j \oplus \mathsf{parity}(z) \oplus \delta$
  end for
  return the majority vote among the b_j's ▷ Predictor for $r \bmod 2$
end function

7.2.2 Equivalence between the (Error-Free) Decisional and Computational Approximate-GCD

In this section, we show the equivalence between the (error-free) decisional and computational Approximate-GCD problems (*cf.* Definition 7.4). As a consequence, it follows directly that the additional noise in the fully homomorphic encryption schemes over the integers of Section 7.1.2 can be removed, simplifying the scheme and the security proof given in [vDGHV10] – *cf.* Section 7.3.1.

Theorem 7.5. *Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$. The computational Error-Free-AGCD of parameters $(\gamma, \eta, \nu, \rho)$ is computationally equivalent to the decisional Error-Free-AGCD of parameters $(\gamma, \eta, \nu, \rho + 2\lambda)$.*

For the proof of the Theorem, we will use the following decisional problem:

Definition 7.6. Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$. The *Error-Free LSB AGCD* problem is: For a 2^ν-rough η-bit integer p and a uniformly chosen 2^ν-rough $q_0 \in [0, 2^\gamma/p)$, given $x_0 = q_0 \cdot p$ and polynomially many samples from $D_\rho(p, q_0)$, determine $b \in \{0, 1\}$ from $z = q \cdot p + 2r + b \cdot c$ where $q \leftarrow [0, q_0)$, $r \leftarrow \mathbb{Z} \cap [0, 2^{\rho-1})$ and $c \leftarrow \{0, 1\}$.

Proof. One can show that the computational EF-AGCD and the Error-Free LSB AGCD problems are computationally equivalent. Indeed, we can construct a high-accuracy LSB predictor subroutine (*cf.* Algorithm 7.1) using an adversary $\mathcal{A}$ having a non-negligible advantage ε against the $(\gamma, \eta, \nu, \rho')$-Error-Free LSB AGCD problem (with $\rho' \geqslant \rho + 2\lambda$),[5] and by using it in Step 2 of the security proof of [vDGHV10], we automatically get the equivalence.

Let us show that the decisional EF-AGCD and Error-Free LSB AGCD problems are computationally equivalent (one of the reductions is trivial). We consider the sequence of distributions for $\rho \leqslant i \leqslant \eta + \lambda$:

$$D'_\rho(p, q_0, i) = \left\{ q \cdot p + 2^{\lambda+\eta-i} \cdot r \bmod (q_0 p) : q \leftarrow [0, q_0), r \leftarrow \mathbb{Z} \cap [0, 2^i) \right\} .$$

Note that in the distribution $D'_\rho(p, q_0, i)$ above the size of the random r is i-bit instead of ρ-bit. For $i = \rho$, the distribution of y is the same as the distribution $D_\rho(p, q_0)$, up to a factor $2^{\lambda+\eta-\rho}$ modulo x_0. One can show that for $i = \eta + \lambda$, the distribution $D'_\rho(p, q_0, i)$ is $2^{-\lambda}$-statistically close to uniform modulo x_0. Therefore by a standard hybrid argument, if a distinguisher solves the decisional EF-AGCD problem with some non-negligible advantage, then he must be able to distinguish between two successive distributions $D'_\rho(p, q_0, i)$ and $D'_\rho(p, q_0, i + 1)$ for some i.

[5] The additional noise is used to drown the noise due to the public key elements and z.

Let us consider the challenge from the Error-Free LSB AGCD problem:

$$z = q \cdot p + 2r + b \cdot c$$

where $r \leftarrow \mathbb{Z} \cap [0, 2^{\rho-1})$ and $c \leftarrow \{0, 1\}$. We let:

$$y = 2^{\lambda+\eta-i-1} \cdot (2^{\rho} \cdot u + z) \bmod x_0$$

where $u \leftarrow \mathbb{Z} \cap [0, 2^{i+1-\rho})$. This gives:

$$\begin{aligned} y &= q' \cdot p + 2^{\lambda+\eta-i-1} \cdot (2^{\rho} \cdot u + 2r + b \cdot c) \bmod x_0 \\ &= q' \cdot p + 2^{\lambda+\eta-i-1} \cdot (2r' + b \cdot c) \end{aligned}$$

for some $q' \in \mathbb{Z}$ uniformly distributed in $[0, q_0)$ (with overwhelming probability and because q is uniformly distributed in $[0, q_0)$), where $r' \leftarrow \mathbb{Z} \cap [0, 2^i)$.

If $b = 0$ then we get $y = q' \cdot p + 2^{\lambda+\eta-i} \cdot r'$ which corresponds to the distribution $D'_{\rho}(p, q_0, i)$. If $b = 1$ then we get $y = q' \cdot p + 2^{\lambda+\eta-i-1} \cdot r''$ where $r'' \leftarrow \mathbb{Z} \cap [0, 2^{i+1})$, which corresponds to the distribution $D'_{\rho}(p, q_0, i+1)$. Therefore we can use the previous distinguisher to solve the Error-Free LSB AGCD problem. □

7.2.3 An AGCD Distribution with Several Primes

Let us extend the definition of the AGCD distribution for several primes. This will be useful to build a semantically secure multi-slot DGHV scheme in Section 7.3.2.

Definition 7.7. Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$ and $\ell \in \mathbb{Z}$, $\ell \geqslant 1$. The *ℓ-AGCD distribution* $D^{(\ell)}_{\rho}(p_1, \ldots, p_\ell, q_0)$, for given coprime 2^{ν}-rough η-bit integers $p_1, \ldots, p_\ell$ and a uniformly chosen 2^{ν}-rough $q_0 \in [0, 2^{\gamma}/\pi)$ coprime with π, where $\pi = p_1 \times \cdots \times p_\ell$, is the set of integers

$$x_i = \mathsf{CRT}_{q_0, p_1, \ldots, p_\ell}(q_i, r_{i1}, \ldots, r_{i\ell})$$

where $q_i \in [0, q_0)$ and $r_{ij} \in [0, 2^{\rho})$ are sampled uniformly. The *computational-AGCD* problem is: For a 2^{ν}-rough η-bit integer p and a uniformly chosen 2^{ν}-rough $q_0 \in [0, 2^{\gamma}/p)$, given polynomially many samples from $D^{(\ell)}_{\rho}(p, q_0)$, to compute p.

We have the following lemma:

Lemma 7.8. *The 1-AGCD distribution $D^{(1)}_{\rho}(p_1, q_0)$ is statistically indistinguishable from the AGCD distribution $D_{\rho}(p_1, q_0)$.*

Proof. Let us consider an AGCD sample $x = q \cdot p_1 + r \leftarrow D_{\rho}(p_1, q_0)$, where q is uniformly generated in $[0, q_0)$ and r is uniformly generated in $[0, 2^{\rho})$.

With overwhelming probability we have that p_1 and q_0 are coprime (since $\nu > \lambda$, *cf.* Table 7.1). Therefore, we can write

$$x = \mathsf{CRT}_{q_0, p_1}(x \bmod q_0, x \bmod p_1) = \mathsf{CRT}_{q_0, p_1}(q', r)\,,$$

for an integer q'. Since q is uniform modulo q_0, we have that $q' = (q \cdot p_1 + r) \bmod q_0$ is uniformly distributed in $[0, q_0)$ with overwhelming probability, which concludes the proof. □

Next, let us define a new decisional problem:

Definition 7.9. Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$. The *decisional ℓ-Error-Free-AGCD* problem is: For ℓ coprime 2^{ν}-rough η-bit integers $p_1, \ldots, p_\ell$ and a uniformly chosen 2^{ν}-rough $q_0 \in [0, 2^{\gamma}/\pi)$ coprime with π, where $\pi = p_1 \times \cdots \times p_\ell$, given $x_0 = q_0 \cdot \pi$ and polynomially many samples from $\mathbb{Z}_{x_0}$ to distinguish whether the samples are distributed uniformly or whether they are distributed according to the ℓ-AGCD distribution $D^{(\ell)}_{\rho}(p_1, \ldots, p_\ell, q_0)$.

We have the following immediate lemma:

Lemma 7.10. *Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$. The decisional 1-EF-AGCD is hard under the decisional EF-AGCD assumption.*

Proof. The 1-EF-AGCD problem is the problem to distinguish the uniform distribution modulo $x_0 = q_0 \cdot p_1$ from $D_\rho^{(1)}(p_1, q_0)$. Now, by Lemma 7.8, $D_\rho^{(1)}(p_1, q_0)$ is indistinguishable from $D_\rho(p_1, q_0)$, and under the EF-AGCD assumption, it is intractable to distinguish between the distribution $D_\rho(p_1, q_0)$ and the uniform distribution modulo x_0; the result follows. □

Finally, we have an interesting reduction:

Lemma 7.11. *Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$. The decisional ℓ-EF-AGCD with parameters $(\gamma, \eta, \nu, \rho)$ is hard under the decisional 1-EF-AGCD assumption with parameters $(\gamma - (\ell-1)\eta, \eta, \nu, \rho)$.*

Combining Lemmas 7.10 and 7.11, we have the immediate corollary:

Corollary 7.12. *Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$. The decisional ℓ-EF-AGCD with parameters $(\gamma, \eta, \nu, \rho)$ is hard under the decisional EF-AGCD assumption with parameters $(\gamma - (\ell-1)\eta, \eta, \nu, \rho)$.*

Proof of Lemma 7.11. Assume we have an adversary $\mathcal{A}$ having non-negligible advantage ε against the ℓ-EF-AGCD problem. We show that we can construct a polynomial-time algorithm having advantage ε/ℓ against the 1-EF-AGCD problem.

$\mathcal{B}$ has access to $x_0 = q_0 \cdot p_1$ and samples $\{y_i\}$ of $\mathbb{Z}_{x_0}$ either uniformly distributed in $\mathbb{Z}_{x_0}$, either from $D_\rho^{(1)}(p_1, q_0)$ and has to guess which is the case.

First she selects $i_0 \in \{1, \ldots, \ell\}$, the position of the unknown prime p_1. First, $\mathcal{B}$ picks $\ell - 1$ coprime 2^ν-rough integers $p_2, \ldots, p_\ell$. With overwhelming probability (because $\nu > \lambda$) $p_2, \ldots, p_\ell$ are coprime with p_1 and q_0. Next she defines $x'_0 = x_0 \times p_2 \times \cdots \times p_\ell$, and send it to $\mathcal{A}$.

When $\mathcal{A}$ asks a sample, $\mathcal{B}$ gets a sample y_i and computes

$$x_i = \mathsf{CRT}_{x_0, p_{((1-i_0) \bmod \ell)+1}, \ldots, p_\ell, p_2, \ldots, p_{((\ell-i_0) \bmod \ell)+1}}(y, r'_{i1}, \ldots, r'_{i(i_0-1)}, r_{i(i_0+1)}, \ldots, r_{i\ell})$$

where $r_{i(i_0+1)}, \ldots, r_{i\ell} \in [0, 2^\rho)$ and $r'_{ik} \in [0, p_k)$ for $k = 1, \ldots, i_0 - 1$ are sampled uniformly, and transmits it to $\mathcal{A}$.

For all $j = 0, \ldots, \ell$, define D_j the set of integers

$$\mathsf{CRT}_{q_0, p_{((1-i_0) \bmod \ell)+1}, \ldots, p_\ell, p_1, \ldots, p_{((\ell-i_0) \bmod \ell)+1}}(q_i, r'_{i1}, \ldots, r'_{ij}, r_{i(j+1)}, \ldots, r_{i\ell})$$

where $q_i \in [0, q_0)$, $r_{i(j+1)}, \ldots, r_{i\ell} \in [0, 2^\rho)$ and $r'_{ik} \in [0, p_k)$ for $k = 1, \ldots, j$ are sampled uniformly. In particular, we have

$$D_0 = D_\rho^{(\ell)}(p_{((1-i_0) \bmod \ell)+1}, \ldots, p_\ell, p_1, \ldots, p_{((\ell-i_0) \bmod \ell)+1}, q_0)$$

and D_ℓ is the uniform distribution over $[0, x_0)$. By a standard hybrid argument, we know that $\mathcal{A}$ has at least advantage ε/ℓ to distinguish between D_{j-1} and D_j for a $j \in \{1, \ldots, \ell\}$. Therefore with probability $1/\ell$, we have $i_0 = j$. In that case, $\mathcal{B}$ sends to $\mathcal{A}$ samples from D_{i_0-1} when the y_i's were sampled from $D_\rho^{(1)}(p_1, q_0)$, and from D_{i_0} when the y_i's were uniformly distributed in $[0, x_0)$.[6]

Finally, $\mathcal{B}$ can use the answer of $\mathcal{A}$ to the ℓ-EF-AGCD problem to solve the 1-EF-AGCD problem with advantage ε/ℓ. □

7.2.4 Attacks and Parameters Derivation

In this section, we briefly review the attacks of [HG01, vDGHV10, CMNT11, CNT12, CN12, CH12] on the EF-AGCD problem, and use them to derive parameters for λ bits of security. Assume that we have a $(\gamma, \eta, \nu, \rho)$-EF-AGCD instance, with $x_0 = q_0 \cdot p$ and polynomially many $x_i = q_i \cdot p + r_i$, and that we want to recover p.

[6] This is due to the associativity of the CRT; namely that, using simplified notation, $\mathsf{CRT}(a, b, c) = \mathsf{CRT}(\mathsf{CRT}(a, b), c) = \mathsf{CRT}(a, \mathsf{CRT}(b, c))$.

Removing the Noise. A first idea is to try to remove the noise from, say, the sample x_1. Note that, since q_0 is 2^ν-rough and q_1 is uniformly distributed in $[0, q_0)$, we have that $\gcd(q_0, q_1) = 1$ with overwhelming probability (because $\nu > \lambda$), *i.e.* $\gcd(x_0, x_1 - r_1) = p$ with overwhelming probability. A simple technique is therefore to try to guess r_1 and verify the guess with a GCD computation [vDGHV10]. Specifically, for $\hat{r}_1 \in [0, 2^\rho)$, set $\hat{x}_1 = x_1 - \hat{r}_1$ and $\hat{p} = \gcd(x_0, \hat{x}_1)$; if $\hat{p}$ has η bits output $\hat{p}$ as a possible solution. Obviously, p will be found by this technique in less than 2^ρ tries, and for the parameter choices where ρ is much smaller than η, the solution is likely to be unique. Finally the complexity of this attack is $\mathcal{O}(2^\rho)$.

In [CN12], Chen and Nguyen presented a new algorithm to solve EF-AGCD of complexity $\mathcal{O}(2^{\rho/2})$, that is exponentially faster. The key idea is to realize that

$$p = \gcd\left(x_0, \prod_{\hat{r}_1=0}^{2^\rho-1} (x_1 - \hat{r}_1) \bmod x_0\right). \tag{7.3}$$

Indeed,

$$\prod_{\hat{r}_1=0}^{2^\rho-1} (x_1 - \hat{r}_1) \bmod x_0 = [(q_1 \cdot Q) \cdot p] \bmod (q_0 \cdot p)$$

with $Q = \prod_{\hat{r}_1 \neq r_1} (x_1 - \hat{r}_1)$, and since q_1 is uniformly distributed modulo q_0, $q_1 \cdot Q$ is uniformly distributed modulo q_0 and Equation (7.3) is verified with overwhelming probability. Applied naively, this technique still yields an attack of complexity $\mathcal{O}(2^\rho)$.[7] However Equation (7.3) can be exploited in a much more powerful way. Define the polynomial $f_j(x)$ of degree j, with coefficients modulo x_0:

$$f_j(x) = \prod_{i=0}^{j-1} (x_1 - (x + i)) \pmod{x_0},$$

and set $\rho' = \lfloor \rho/2 \rfloor$ and $\epsilon = (\rho \bmod 2) = \rho - 2\rho'$. It follows that

$$p = \gcd\left(x_0, \prod_{\hat{k}=0}^{2^{\rho'+\epsilon}-1} f_{2^{\rho'}}\left(k \cdot 2^{\rho'}\right) \bmod x_0\right). \tag{7.4}$$

To compute Equation (7.4), one is reduced to perform one GCD, $(2^{\rho'+\epsilon} - 1)$ modular multiplications, and a multi-evaluation of a polynomial $f_{2^{\rho'}}$ at $2^{\rho'+\epsilon}$ points. Now computing the polynomial $f_{2^{\rho'}}$ itself costs $\tilde{\mathcal{O}}(2^{\rho'})$ modular multiplications, using a tree structure to multiply polynomials of the same degree (see [CN12, Alg. 2]), and the multi-evaluation of $f_{2^{\rho'}}$ costs $\tilde{\mathcal{O}}(2^{\rho'})$ modular operations using Fast Fourier techniques (the key idea being that $f_{2^{\rho'}}(\alpha) = f_{2^{\rho'}}(x) \bmod (x - \alpha)$, and this can be done efficiently using a tree structure; see [CN12, Alg. 3]). Using the underlying structure in the factors of $f_{2^{\rho'}}(x)$, a logarithmic speedup can be obtained [CN12, Sec. 2.3] and the time complexity of the whole attack is $\mathcal{O}(2^{\rho'})$ modular multiplications, and requires $\mathcal{O}(2^{\rho'})$ memory.[8]

Deriving Parameters. This attack gives a condition on ρ and γ to fix parameters. Since a modular multiplication with a γ-bit modulus has complexity $\mathcal{O}(\gamma \log_2 \gamma)$, to ensure λ bits of security, we have the following condition:

$$C \cdot 2^{\rho'} \cdot (\gamma \log_2 \gamma) \geqslant 2^\lambda,$$

for a constant C, that is

$$\rho \geqslant 2(\lambda - \log_2 \gamma - \log_2 \log_2 \gamma - \log_2 C).$$

Let us estimate the constant C such that the attack requires $C \cdot 2^{\rho/2} \cdot \gamma \log_2 \gamma$ cycles with unbounded memory.[9] We can estimate a lower bound on C from the running times of the attacks

[7] But with a smaller constant term as it performs 2^ρ modular multiplications and one GCD-computation instead of 2^ρ GCD-computations [CMNT11].

[8] A time-memory trade-off can be used (see [CN12, Sec. 3.1]) and yields an attack using $\tilde{\mathcal{O}}(2^\rho/d)$ modular operations and $\tilde{\mathcal{O}}(d)$ memory for $d \leqslant 2^{\rho/2}$.

[9] We use the security level definition given in [CMNT11], that states that a scheme has λ bits of security if the best attack against it takes at least 2^λ cycles.

on the Toy parameters (*cf.* [CN12]), and get that

$$C \geqslant \frac{(2.27 \cdot 10^9) \cdot 99}{(2^{17}/2^8) \cdot (1.6 \cdot 10^5) \cdot \log_2(1.6 \cdot 10^5)} \approx 158\,.$$

Thus, to obtain $\lambda = 72$ bits of security with $\gamma \approx 2^{25}$, it suffices to take $\rho \geqslant 70$ and this is coherent with the parameters of [CMNT11, CNT12, CCK+13, CLT13b].

Remark 7.13. Note that the previous condition is somewhat tight (namely the constants in the $\mathcal{O}$ have *not* been ignored, but an unbounded memory is assumed to be available to the adversary). To be more conservative, one could for example select arbitrarily $C = 1$ (or any other value smaller than 158), which increases the values of ρ by about 14 bits. To be less conservative, one could select parameters taking into account the time-memory trade-off described in [CN12, Sec. 3.1].

We provide a small `SAGE` [S+14] function to estimate the cost of this attack on Figure 7.1.

```
def cost_CN12_attack(sec_parameter,gamma,C=158):
    return RR(2*(sec_parameter-log(gamma, base=2)-log(log(gamma, base=2),
    base=2)-log(C, base=2)))
```

Figure 7.1 – `SAGE` function to estimate the cost of Chen and Nguyen's attack.

Continued Fractions. Using the continued fractions of a real number x, one can derive a sequence of pairs of integers (s_i, t_i) such that $|x - s_i/t_i| < 1/t_i^2$; we say that s_i/t_i is a convergent of x. Now if $x = x_i/x_j \approx q_i/q_j$ for some i, j, then q_i/q_j seems a good approximation of x, and one might hope it is a convergent, and therefore to recover it using continued fractions. This would immediately yields $p = \lfloor x_i/q_i \rceil$.

Let $i \neq j$ be integers. Denote $x = x_i/x_j$ and $d = \gcd(q_i, q_j)$. There exists coprime integers q_i', q_j' such that $q_i = dq_i'$ and $q_j = dq_j'$, and we have

$$\left| x - \frac{q_i}{q_j} \right| = \left| \frac{x_i}{x_j} - \frac{q_i'}{q_j'} \right| = \left| \frac{q_j' r_i - q_i' r_j}{q_j'(q_j' dp + r_j)} \right| = \left| \frac{q_j' r_i - q_i' r_j}{dp + r_j/q_j'} \right| \cdot \frac{1}{{q_j'}^2}\,. \tag{7.5}$$

Applying the previous argument, we will recover dp (instead of p) from the knowledge of q_i' or q_j' (if they are large enough). Let us denote $\delta = \lceil \log_2 d \rceil$; with overwhelming probability, we have $\delta \leqslant \lambda$, and therefore using ECM, one will easily recover p from dp. Indeed, the probability that two random integers have GCD equals to i is $i^{-2}/\zeta(2)$ (where ζ is the Riemann zeta function) [CS87], therefore

$$\Pr[\gcd(q_i, q_j) > 2^\lambda] = 1 - \sum_{i=1}^{2^\lambda} \frac{i^{-2}}{\zeta(2)}\,.$$

Now

$$\sum_{i=1}^{2^\lambda} i^{-2} = \sum_{i=1}^{\infty} i^{-2} - \sum_{i=2^\lambda+1}^{\infty} i^{-2} \leqslant \zeta(2) - \int_{2^\lambda+1}^{\infty} i^{-2}\mathrm{d}i = \zeta(2) - \frac{1}{2^\lambda + 1}\,.$$

Thus $\Pr[\gcd(q_i, q_j) > 2^\lambda]$ is negligible in the security parameter λ.

Finally, in order to avoid that one might recover q_i'/q_j' from the continued fraction of x, it suffices that

$$\left| \frac{q_j' r_i - q_i' r_j}{dp + r_j/q_j'} \right| \gg 1\,.$$

Now with overwhelming probability $q_0 > 2^{\gamma-\eta-\lambda}$, and thus $q_i, q_j > 2^{\gamma-\eta-2\lambda}$, and thus the numerator will be larger than $2^{\gamma-\eta-3\lambda}$. Since the denominator is at most equal to $2^{\lambda+\eta+1}$, as long as $2\eta+5\lambda < \gamma$, with overwhelming probability the adversary will not recover an useful approximation of x from its continued fraction approximations and the attack does not apply. Below, we will obtain much more efficient attacks by working in higher dimensional lattices.

Remark 7.14. Note that having an exact multiple of p, namely $x_0 = q_0 \cdot p$, corresponds to the case where $i = 0$ (and $r_i = 0$); in particular it does not improve the previous attack.

Lattice Attacks. Three main attacks were proposed to attack the Approximate-GCD problem: a generalization of the continued fraction approach with Coppersmith's techniques [Cop97] by Cohn and Heninger [CH12], a simultaneous Diophantine equation approach [vDGHV10] and a orthogonal lattice attack [vDGHV10, CMNT11].

Coppersmith's Attack. In [HG01], Howgrave-Graham proposed to use lattices to solve the EF-AGCD problem with two elements. His approach was generalized in 2012 by Cohn and Heninger for the general version of the EF-AGCD problem [CH12]. In a nutshell, the EF-AGCD problem is closely related to the problem of finding small roots of multivariate polynomials using powerful lattice-based techniques due to Coppersmith [Cop97]. For all $i = 1, \ldots, m$, denote $x_i = q_i \cdot p + r_i$. The approach consists in constructing multivariate polynomials $y_1, \ldots, y_m \in \mathbb{Z}[t_1, \ldots, t_m]$ having $\mathbf{r} = (r_1, \ldots, r_m)^t$ as root, *i.e.* such that

$$y_i(r_1, \ldots, r_m) = 0\,, \tag{7.6}$$

and to solve the system of equations to recover the r_i's. We will not be able to construct directly a y_i such that Equation (7.6) holds over $\mathbb{Z}$, but we can obtain such an equality modulo a power of p. Then using lattice reduction, one might obtain a polynomial y with small coefficients, and since the r_i's are small hope that Equation (7.6) holds over $\mathbb{Z}$.

Let denote k (to be chosen later) and construct vectors y's such that $y(r_1, \ldots, r_m) = 0 \bmod p^k$ and such that $|y(r_1, \ldots, r_m)| < 2^{(\eta-1)\cdot k}$, so that $y(r_1, \ldots, r_m) = 0$ over $\mathbb{Z}$ using the fact that $p \geqslant 2^{\eta-1}$. The approach of [HG01, CH12] is to construct such small y's as integer combinations of the shift polynomials

$$(x_1 - t_1)^{i_1} \cdots (x_m - t_m)^{i_m} x_0^{\ell}\,,$$

for $i_1 + \cdots + i_m + \ell \geqslant k$, *i.e.* a short vector of the integer lattice generated by the coefficient vectors of these polynomials. Note that denoting such a polynomial

$$y(t_1, \ldots, t_m) = \sum_{j_1, \ldots, j_m} a_{j_1, \ldots, j_m} t_1^{j_1} \cdots t_m^{j_m}\,,$$

we have that

$$|y(r_1, \ldots, r_m)| \leqslant \sum_{j_1, \ldots, j_m} |a_{j_1, \ldots, j_m}| 2^{\rho(j_1 + \cdots + j_m)} = \|\mathbf{a}\|_1\,, \tag{7.7}$$

where $\mathbf{a}$ has entries $a_{j_1, \ldots, j_m} 2^{\rho(j_1 + \ldots + j_m)}$. Thus every vector $\mathbf{a}$ such that $\|\mathbf{a}\|_1 < 2^{(\eta-1)\cdot k}$ gives a polynomial relation between the r_i's over $\mathbb{Z}$. Therefore we incorporate the bounds on the r_i's into the lattice to find a y small enough: we use the lattice L generated by the coefficient vectors of the polynomials

$$(2^{\rho} \cdot x_1 - t_1)^{i_1} \cdots (2^{\rho} \cdot x_m - t_m)^{i_m} x_0^{\ell}\,,$$

with $i_1 + \cdots + i_m \leqslant T$ and $\ell = \max(k - i_1 - \cdots - i_m, 0)$, where T, k are parameters to be chosen later. The dimension of the lattice L is equal to the number of monomials of degree at most T in m unknowns, *i.e.*

$$\dim(L) = \binom{T + m}{m}\,.$$

Moreover, using a monomial ordering so that the basis of the matrix is upper triangular, we have

$$\det(L) = (2^{\rho})^{s_{\rho}} x_0^{s_{x_0}}\,,$$

where s_{ρ} is the sum of the exponents of all unknowns in all occurring polynomials and s_{x_0} is the sum of exponents of x_0 in all occurring polynomials. The number of polynomials of degree d in m unknowns is $\binom{d+m-1}{m-1}$, thus

$$\begin{aligned} s_{\rho} &= \sum_{d=0}^{T} d \cdot \binom{d + m - 1}{m - 1} = \sum d \frac{(d + m - 1)!}{d!(m - 1)!} = \sum \frac{(d - 1 + m)! m}{(d - 1)! m!} \\ &= m \sum_{d=0}^{T-1} \binom{d + m}{m} = m \sum_{d=m}^{T+m-1} \binom{d}{m} = m \cdot \binom{T + m}{m + 1} = m \cdot \frac{(T + m)!}{(m + 1)!(T - 1)!} \\ &= \frac{mT}{m + 1} \binom{T + m}{m}\,, \end{aligned}$$

and

$$s_{x_0} = \sum_{d=0}^{k}(k-d)\cdot\binom{d+m-1}{m-1}k\binom{k+m}{m} - \frac{mk}{m+1}\binom{k+m}{m} = \frac{km+k-mk}{m+1}\binom{k+m}{m}$$
$$= \frac{k}{m+1}\binom{k+m}{m}.$$

Therefore, for the attack to be successful, we need to have

$$\det(L)^{1/\binom{T+m}{m}} \leqslant 2^{k\cdot(\eta-1)},$$

therefore that

$$\frac{k}{m+1}\cdot\binom{k+m}{m}\cdot\log_2(x_0) \leqslant k\cdot(\eta-1)\cdot\binom{T+m}{m},$$

and then that

$$\binom{T+m}{m} \geqslant \frac{\gamma-\lambda}{\eta-1},$$

because $\log_2(x_0) > \gamma-\lambda$ with overwhelming probability and $\binom{k+m}{m} \geqslant k(m+1)$ when $m \geqslant 2$.

Now, a classical lattice 'rule of thumb' conjecture states that there exists an absolute constant $c > 1$ such that for any K, any M and any sufficiently regular M-dimensional lattice, one cannot find a $c^{M/K}$ approximation of the shortest vector in time smaller than 2^K. Therefore to recover a $2^{\eta-1}$ approximation of the short vector of L (which is not even sufficient for the attack to succeed) with $\binom{T+m}{m} \geqslant \frac{\gamma-\lambda}{\eta-1}$ the time required is then at least $2^{(\gamma-\lambda)/(\eta-1)^2\cdot\log_2 c}$, which yields the asymptotic (conservative) following condition to resist the attack:

$$\gamma = \eta^2\cdot\Omega(\lambda).$$

Simultaneous Diophantine Equations. This attack is introduced in [vDGHV10], and is based on the fact that the rationals $y_i = x_i/x_0$ are an instance of the simultaneous Diophantine approximation. Indeed, we have that $\frac{x_i}{x_0} = \frac{q_i+s_i}{q_0}$ with $|s_i| = \left|\frac{r_i}{p}\right| \leqslant 2^{\rho-(\eta-1)}$ (because $p > 2^{\eta-1}$). We want to find a vector $\mathbf{v} = (q, v_1, \ldots, v_m)^t$ such that

$$\left(q\frac{x_1}{x_0} - v_1, \ldots, q\frac{x_n}{x_0} - v_n\right)^t,$$

has a minimal norm. This is achieved by the vector $\mathbf{v}_0 = (q_0, q_1, \ldots, q_m)^t$, which would allow to recover $p = x_0/q_0$.

Therefore we can try to use Lagarias' algorithm [Lag82] to find an approximation of q_0 using lattice reduction algorithms. Let us consider the matrix

$$\mathbf{B}^t = \begin{pmatrix} 2^\rho & x_1 & x_2 & \cdots & x_t \\ & -x_0 & & & \\ & & -x_0 & & \\ & & & \ddots & \\ & & & & -x_0 \end{pmatrix}.$$

Our target solution corresponds to a vector of norm smaller than $\sqrt{m+1}\cdot 2^{\gamma+\rho-\eta+1}$, specifically

$$\mathbf{v}^t = \mathbf{v}_0^t\cdot\mathbf{B}^t = \left(q_0\cdot 2^\rho, \quad q_0x_1 - x_0q_1, \quad \cdots, \quad q_0x_m - x_0q_m\right),$$

where $|q_0x_i - q_ix_0| = |q_0r_i| \leqslant 2^{\gamma-(\eta-1)+\rho}$ for all i, and $|q_0\cdot 2^\rho| \leqslant 2^{\gamma-(\eta-1)+\rho}$. However this vector $\mathbf{v}$ is not necessarily the shortest non-zero vector in the lattice, and thus might not be recovered by lattice reduction. Indeed, by Minkowski's bound we expect the shortest vector to be of size *at most* $\sqrt{m+1}\cdot\det(\mathbf{B}^t)^{1/(m+1)} < \sqrt{m+1}\cdot 2^{(\rho+\gamma\cdot m)/(m+1)}$.

Let us give an upper bound on the values m for which this vector is shorter than $\|\mathbf{v}\|$. With overwhelming probability, we have $q_0 > 2^{\gamma-\eta-\lambda}$, and therefore $\|\mathbf{v}\| \geqslant \sqrt{m+1}\cdot 2^{\gamma-\eta-\lambda}$. Assume

that m is such that $\sqrt{m+1} \cdot 2^{\gamma-\eta-\lambda} > \sqrt{m+1} \cdot 2^{(\rho+\gamma\cdot m)/(m+1)}$; in particular $\mathbf{v}$ is not the shortest non-zero vector in the lattice. We get that

$$\gamma - \eta - \lambda > \frac{\rho + m \cdot \gamma}{m+1}\,,$$

then that

$$(m+1)\gamma - (m+1)(\eta+\lambda) > \rho + m \cdot \gamma\,,$$

and thus

$$\gamma - \rho > (m+1) \cdot (\eta + \lambda)\,,$$

i.e. $m+1 < (\gamma-\rho)/(\eta+\lambda)$. Heuristically, B will tend to have exponentially (in m) many vectors which obscure our target solution.

In order to $\mathbf{v}$ to be possibly the shortest vector of the lattice, we now assume that $m+1 \geqslant \gamma'/\eta'$ where $\gamma' = \gamma - \rho$ and $\eta' = \eta + \lambda$. Now when m is large, lattice reduction will not recover the short vector $\mathbf{v}$ but only an approximation. By the classical lattice 'rule of thumb' conjecture cited below, there exists an absolute constant $c > 1$ such that for any k, any m and any sufficiently regular m-dimensional lattice, one cannot find a $c^{m/k}$ approximation of the shortest vector in time smaller than 2^k. Therefore to recover a $2^{\eta'}$ approximation of the short vector of L (which is not even sufficient to recover q_0) with $m+1 \geqslant \gamma'/\eta'$ the time required is then at least $2^{\gamma'/\eta'^2 \cdot \log_2 c}$, which yields the asymptotic (conservative) following condition to resist the attack:

$$\gamma = \eta^2 \cdot \Omega(\lambda)\,.$$

Orthogonal Lattice Attacks. A promising lattice-based attack against the EF-AGCD problem was proposed in [vDGHV10] (and more precisely assessed in [CMNT11] to derive parameters) and is based on Nguyen and Stern's orthogonal lattice [NS01]. Let us consider m integers $x_1, \ldots, x_m$. Consider a vector $\mathbf{u}$ orthogonal to $\mathbf{x} = (x_1, \ldots, x_m)^t$ modulo x_0, that is such that

$$\sum_{i=1}^{m} u_i \cdot x_i = 0 \bmod x_0\,.$$

This gives

$$\sum_{i=1}^{m} u_i \cdot r_i = 0 \bmod p\,,$$

and when the u_i's are small enough, namely such that $\|\mathbf{u}\|_\infty \leqslant 2^{\eta-1-\rho-\log_2 m}$, since the r_i's are smaller than 2^ρ, the latter equality will hold over $\mathbb{Z}$. By recovering many such small vectors $\mathbf{u}$ orthogonal to $\mathbf{r} = (r_1, \ldots, r_m)^t$ over $\mathbb{Z}$, one can recover $\mathbf{r}$ easily. Therefore, we will consider this attack to be successful once one such small vector $\mathbf{u}$ orthogonal to $\mathbf{r}$ in $\mathbb{Z}$ has been recovered.

Let $L \subset \mathbb{Z}^m$ be the lattice of row vectors orthogonal to $\mathbf{x}$ modulo x_0. Clearly, L contains $x_0\mathbb{Z}^m$ so it is of full rank m. Moreover, we have

$$\det(L) = [\mathbb{Z}^m : L] = x_0 / \gcd(x_0, x_1, \ldots, x_m) = x_0\,.$$

The key idea of the attack is to find a short enough vector $\mathbf{u}$ in the lattice. From Minkowski's bound, one might expect there exists a non-zero lattice vector of norm $\sqrt{m} \cdot \det(L)^{1/m}$ (up to a multiplicative factor). Thus to recover $\mathbf{u}$, this gives the approximate condition $\sqrt{m} \cdot 2^{\gamma/m} \leqslant 2^{\eta-1-\rho-\log_2 m}$ that can be loosened into $2^{\gamma/m} < 2^\eta$, and this yields $m > \gamma/\eta$. As in the previous attack, since m is large, lattice reduction will only recover an approximation of the shortest vector. To recover a 2^η approximation of the short vector of L (which is not even sufficient to recover $\mathbf{u}$) with $m > \gamma/\eta$ the time required is then at least $2^{\gamma/\eta^2 \cdot \log_2 c}$, which yields (again) the asymptotic following condition to resist the attack:

$$\gamma = \eta^2 \cdot \Omega(\lambda)\,.$$

Deriving Parameters. For a fixed η, we can use as in [CMNT11] the orthogonal lattice attack to derive a value for γ. To derive concrete parameters, one can set γ such that the LLL [LLL82] and BKZ-20 [CN11] lattice reduction algorithms[10] running time is greater than 2^λ. Since it is often mentioned in the literature that LLL seems to run much faster in practice than the worst-case theoretical bounds, an estimation of LLL running time was provided to derive the parameters.

This running time estimation was done by running experiments on the following matrix $\mathbf{B}$, which verifies $\mathbf{B} \cdot \mathbf{x} = \mathbf{0} \bmod x_0$ (*i.e.* $\mathbf{B}$ is constituted of row vectors orthogonal to $\mathbf{x}$ modulo x_0):

$$\mathbf{B} = \begin{pmatrix} 1 & & & \frac{-x_1}{x_m} \bmod x_0 \\ & 1 & & \frac{-x_2}{x_m} \bmod x_0 \\ & & \ddots & \vdots \\ & & 1 & \frac{-x_{m-1}}{x_m} \bmod x_0 \\ & & & x_0 \end{pmatrix}.$$

With the lattice reduction algorithm $\mathsf{A} \in \{\mathsf{LLL}, \mathsf{BKZ\text{-}20}\}$, one gets a vector $\mathbf{u}$ of norm $\|\mathbf{u}\| = \delta_\mathsf{A}^m \cdot x_0^{1/m}$. Now with overwhelming probability $x_0 > 2^{\gamma-\lambda}$, thus $\|\mathbf{u}\| \geqslant 2^{(\gamma-\lambda)/m + m\cdot\log_2 \delta_\mathsf{A}}$. To ensure that this attack does not succeed when using the reduction algorithm A, it suffices to ensure that $\eta < (\gamma-\lambda)/m + m\cdot\log_2 \delta_\mathsf{A}$, which yields $\|\mathbf{u}\| > 2^\eta$. Consider the equation $\eta = (\gamma-\lambda)/m + m\cdot\log_2 \delta_\mathsf{A}$, or equivalently

$$m^2 \cdot \log_2 \delta_\mathsf{A} - \eta \cdot m + (\gamma - \lambda) = 0\,. \tag{7.8}$$

Its discriminant is $\Delta = \eta^2 - 4\log_2 \delta_\mathsf{A}(\gamma - \lambda)$. Therefore given η, one could fix γ such that $\Delta < 0$. In particular this gives the condition

$$\gamma \geqslant \lambda + \eta^2/(4\log_2 \delta_\mathsf{A})\,. \tag{7.9}$$

When γ verifies this equation, the attack of this section is thwarted. Remains the question of obtaining the minimal δ_A achievable for a given security parameter with the known algorithms A – and we have that $\delta_\mathsf{LLL} \approx 1.021$ and $\delta_\mathsf{BKZ\text{-}20} \approx 1.013$ [NS06, CMNT11].[11]

However, this latter inequality yields larger values for γ, and one could wish to work with a smaller value for γ (for efficiency reasons). In [CMNT11], the authors allow Δ to be positive under the condition that for the values m such that $\eta \geqslant (\gamma - \lambda)/m + m \cdot \log_2 \delta_\mathsf{A}$, the expected running cost of LLL and BKZ-20 is at least 2^λ cycles. Since LLL and BKZ running times increase with the dimension, one should work with the smallest dimension possible for which $\Delta > 0$, *i.e.*

$$m_{\min}(\gamma) = \frac{\eta^2 - \sqrt{\Delta}}{2\log_2 \delta_\mathsf{A}} = \frac{\eta - \sqrt{\eta^2 - 4\log_2 \delta_\mathsf{A}(\gamma - \lambda)}}{2\log_2 \delta_\mathsf{A}}\,. \tag{7.10}$$

In [CMNT11], the running times of LLL and BKZ-20 (expressed in number of clock cycles) are extrapolated by

$$T_\mathsf{LLL}(m, \gamma) = 0.06 \cdot m^4 \cdot \gamma \quad \text{and} \quad T_\mathsf{BKZ\text{-}20}(m, \gamma) = 0.36 \cdot m^{4.2} \cdot \gamma\,,$$

[10] Bigger block sizes for BKZ were not considered in [CMNT11], following the observation of [GN08] that block sizes greater than 25 yield a running time exponential in the lattice dimension.

[11] While this thesis was being written, a manuscript [DT14] was posted on the Cryptology ePrint Archive that claimed to propose an attack against the Approximate-GCD problem running in polynomial-time as long as $\rho < \eta/2$ (which would annihilate any hope for selecting parameters for fully homomorphic encryption). We received immediately questions about this result by several researchers in the community. In the conclusion of the manuscript, the authors emphasized that it remained "an open problem to theoretically prove that indeed [the] algorithm works", and only ran experiments with very small parameters which were solvable by any lattice-based attack previously described. This attack was not considered as an immediate threat by the community, as it was not clear why the attack would work and it did not seem to break the FHE schemes over the integers – see for example a blog post of Martin Albrecht [Alb14].

Several weeks later, on February 24th 2014, the manuscript [DT14] was quietly revised and a more precise analysis concludes that the attack is thwarted when $\gamma \geqslant \lambda + \eta^2/(4\log_2 \delta_\mathsf{A})$, where A is the underlying lattice reduction algorithm, *i.e. exactly the same condition* as obtained by the orthogonal lattice attack in Equation (7.9). Therefore, our parameter selection does not have to be modified because of this result; and this attack does not seem to give any additional advantage compared to the orthogonal lattice attack.

where $\gamma = \log_2(x_0)$. In [NS06], Nguyen and Stehlé also discussed the running time of LLL using `fplll` [CPS13], but obtained different results than in [CMNT11]. In Section 7.2.5, we provide some additional insight on this running time and explain why [CMNT11] can be considered as 'asymptotically conservative'.

Finally, it suffices to select γ such that

$$T_{\mathsf{LLL}}(m_{\min}(\gamma), \gamma), T_{\mathsf{BKZ\text{-}20}}(m_{\min}(\gamma), \gamma) > 2^{\lambda} .$$

A function in SAGE [S+14] to derive secure parameters under this attack is given on Figure 7.2.

```
def T_LLL(m,gamma):
    return RR(log(0.06*m^4*gamma,base=2))
def T_BKZ20(m,gamma):
    return RR(log(0.36*m^4.2*gamma,base=2))
def m_min(sec_parameter, eta, gamma, hermite_factor):
    result = (eta - sqrt(eta*eta-4*log(hermite_factor,base=2)*
    (gamma-sec_parameter)))/2/log(hermite_factor, base=2)
    if result.is_real() and result>0:
        return ceil(result)
    else:
        return 2^sec_parameter

def gamma_from_orthogonal_attack(sec_parameter,eta,conservative=False):
    gamma = ceil(sec_parameter+eta*eta/4/log(1.012, base=2))
    if conservative == False:
        while gamma>1.:
            gamma /= 1.1
            m1 = m_min(sec_parameter,eta,gamma,1.021) # LLL
            m2 = m_min(sec_parameter,eta,gamma,1.013) # BKZ-20
            if min(T_LLL(m1,gamma),T_BKZ20(m2,gamma))<sec_parameter:
                gamma *= 1.1
                break
    return gamma
```

Figure 7.2 – SAGE function to select γ so that running the orthogonal lattice attack takes at least 2^{λ} cycles.

Revisiting the Parameters with BKZ-2.0. At Asiacrypt 2011, Chen and Nguyen proposed a modification of BKZ using recent progress on lattice reduction [Kan83, GNR10, HPS11], called BKZ-2.0 [CN11] (see also the full version of the paper in [CN13]). In particular these improvements allow to consider blocksizes $\beta \geqslant 50$.

It follows that the previous parameter derivation might yield parameters less secure than expected because parameters were chosen to ensure resistance to LLL and BKZ-20 lattice reduction algorithms only. However, it is important to note that BKZ-2.0 assumes that the input of BKZ-2.0 is already a LLL-reduced basis. Therefore, we can combine the previous approach and take into account the LLL running-time inherent in BKZ-2.0. Several works analyze the behavior of the Hermite factor during BKZ-2.0 (*e.g.* [CN11, vdPS13, LN14a]), but it is commonly believed that a Hermite factor 1.005 ensures long-term security (*i.e.* more than 128 bits of security [NIS11, ECR12]). Therefore, one can take $\delta_{\mathsf{A}} = 1.005$ in Equation (7.10), and estimate the parameters so that $T_{\mathsf{LLL}}(m_{\min}(\gamma), \gamma) > 2^{\lambda}$ using the extrapolation of LLL running times.

A function in SAGE [S+14] to derive secure parameters under this attack is given on Figure 7.3.

Last but not least, note that the parameters proposed in [CMNT11, CNT12] need to be slightly - but *not noticeably* - increased to take into account BKZ-2.0. This can be explained by the fact that running LLL in dimensions so that BKZ-2.0 might be successful, to pre-process the input basis (and during BKZ-2.0 execution), takes nearly as many cycles as a direct attack using LLL.

```
def gamma_from_orthogonal_attack_2(sec_parameter,eta,hermite_factor=1.005,
conservative=False):
    gamma = ceil(sec_parameter+eta*eta/4/log(hermite_factor, base=2))
    if conservative == False:
        while gamma>1.:
            gamma /= 1.1
            m = m_min(sec_parameter,eta,gamma,hermite_factor)
            if T_LLL(m,gamma) < sec_parameter:
                gamma *= 1.1
                break
    return gamma
```

Figure 7.3 – SAGE function to select γ so that running the orthogonal lattice attack (even with BKZ-2.0) takes at least 2^λ cycles.

7.2.5 Estimating the Running Time of LLL

In order to derive concrete parameters that resist to the orthogonal lattice attack, [CMNT11] estimated the running time of LLL (and BKZ-20) when applied on the m-dimensional matrix

$$\mathbf{B} = \begin{pmatrix} 1 & & & \frac{-x_1}{x_m} \bmod x_0 \\ & 1 & & \frac{-x_2}{x_m} \bmod x_0 \\ & & \ddots & \vdots \\ & & 1 & \frac{-x_{m-1}}{x_m} \bmod x_0 \\ & & & x_0 \end{pmatrix},$$

where x_0 and the x_i's are samples from the Error-Free Approximate-GCD problem. In particular, they extrapolated the running time of LLL (expressed in number of clock cycles) by

$$T_{\mathsf{LLL}}(m, \gamma) = 0.06 \cdot m^4 \cdot \gamma \,,$$

where $\gamma = \log_2(x_0)$. However this claimed running time is not coherent with previous experiments on the practical running time of LLL [NS06, CN12] which, at a given dimension, should be quadratic in the entries dimension, *i.e.* in γ.

We ran experiments using `fplll-4.0.4` [CPS13] (similarly as [NS06]) on matrices $\mathbf{B}$ in dimensions $50, 70$ and 90 for different values of γ. The LLL implementation of `fplll` relies on floating-point computations [NS05, NS06] which is the fastest variant of LLL implemented. We observed that the extrapolated running time of [CMNT11] is practically conservative (*cf.* Figure 7.4) and asymptotically very conservative. Using a subset of our experiments, the least square method gives the following running time

$$T_{\mathsf{new}}(m, \gamma) = 0.00127 \cdot m^{3.18} \cdot \gamma^{1.83} \,.$$

In the worst case, the LLL algorithm of [NS05, NS06] has complexity $\mathcal{O}(m^5\gamma^2)$. Now for our lattice basis[12], the number of rounds is m times smaller than in the worst case, and since the binary sizes are decreasing quickly, we obtain an heuristic complexity of $\mathcal{O}(m^3\gamma^2)$ [NS06]. This is coherent with our estimated running time T_{new} and the $\mathcal{O}(m^{3.16}\gamma^{1.68})$ of [CN12].

Limitations of the Extrapolation. Extrapolating LLL running times from experiments on small matrices (dimensions $\leqslant 100$) and relatively small γ (compared to the expected values of γ for fully homomorphic encryption) might be problematic.

[12]Note that our matrix $\mathbf{B}$ is really similar to a matrix sampled from the Goldstein Mayer distribution [GM03]. We ran experiments showing that the hidden structure in the Approximate-GCD samples does not seem to modify the running-time compared to a matrix with a random x_0 of γ bits, and $x_1, \ldots, x_m$ random modulo x_0. Now, a remark in [Ste10] claims that a lattice reduction algorithm ran on a matrix with the latter distribution seems to have the same running-time than ran on a matrix sampled from the Goldstein Mayer distribution.

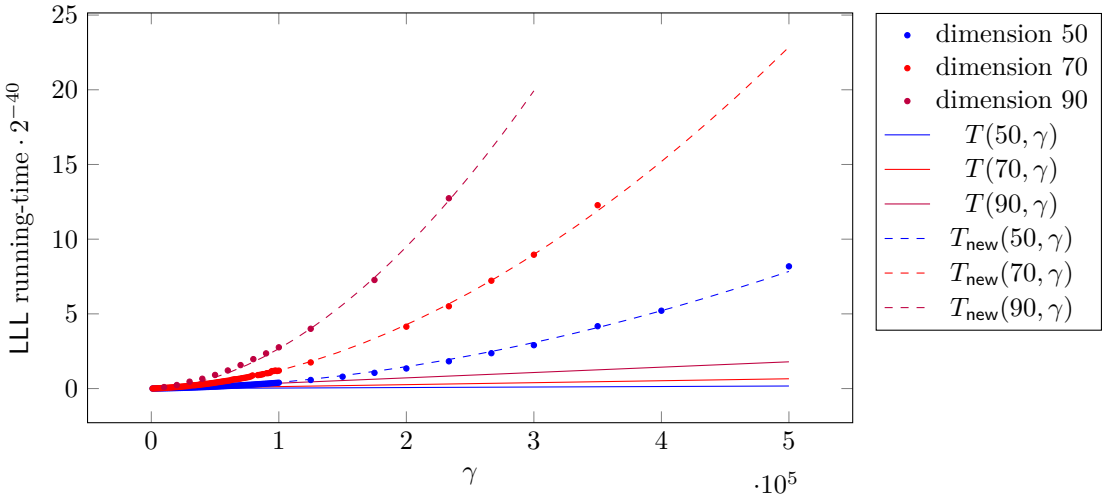


Figure 7.4 – LLL running time in clock cycles using `fplll-4.0.4`

On the bright side, all the experiments mentioned above were realized with double precision (which is extremely fast). In higher dimensions, current implementations of LLL works either with doubles with additional exponents or arbitrary precision floating point numbers which affect *noticeably* the performances of LLL; see [Ste10] for additional information on implementing LLL.

Now, even though the practical experiments described above would suggest consequent improvements in the parameter selection for the Approximate-GCD problem (by replacing T_{LLL} by T_{new}), and therefore yield more efficient FHE schemes, some limitations arise from this approach. Indeed, all the current implementations of LLL and BKZ (including `fplll-4.0.4` [CPS13]) do *not* include the latest improvements in lattice reduction algorithms [NSV11, CN11]. In a way, it might be dangerous to base the choice of the parameters on a polynomial-time problem (LLL running time), as illustrated by possible algorithmic improvements [Ste10, NSV11], and by the fact that optimized implementations might yield a non-negligible gain in the running time.[13]

7.3 Batching the DGHV Scheme

In this section, we revisit the DGHV scheme of [vDGHV10] described in Section 7.1.2 in the error-free setting (which simplifies the scheme and the noise growth) and with a semantic security based on the decisional Approximate-GCD assumption introduced in Section 7.2. Then we extend this one-slot scheme into a multi-slot variant, whose security still relies on the same decisional assumption according to Section 7.2.3. Finally, we briefly discuss the advantages of the multi-slot variant compared to the one-slot scheme in Section 7.3.4.

7.3.1 One-Slot DGHV Scheme

In this section, we revisit and simplify the DGHV scheme, recalled in Section 7.1.2, using an exact multiple x_0 of the secret p and using the equivalence between the decisional EF-AGCD and computational EF-AGCD problems described in Section 7.2.2.

Assume the message space to be $\mathbb{Z}_g$, where g is an α-bit integer. Note that this scheme can be made fully homomorphic only when $g = 2$, following the approaches described in [vDGHV10, CMNT11] and Section 7.4. We present below a somewhat homomorphic encryption scheme, *i.e.* a scheme that handles a limited number of homomorphic operations, where the plaintexts are added or multiplied modulo g.

[13] Novocin, Stehlé and Villard [NSV11] describe a variant of LLL with quasi-linear time complexity in γ. No implementation of this new variant of LLL is publicly available currently; when it will be the case, it might be needed - but not certain as [CMNT11] extrapolation of LLL running-time is already linear in γ - to revisit the DGHV parameters.

$\mathsf{DGHV.Keygen}(1^\lambda, g)$: Generate a 2^ν-rough η-bit integer p and randomly generate a 2^ν-rough integer $q_0 \in [0, 2^\gamma/p)$. Denote $x_0 = p \cdot q_0$. Next sample τ integers $\{x_i\}_{i=1,\ldots,\tau}$ from the AGCD distribution $D_\rho(p, q_0)$.

Let $\mathsf{sk} = p$ and $\mathsf{pk} = \{x_0, x_1, \ldots, x_\tau\}$.

$\mathsf{DGHV.Encrypt}(\mathsf{pk}, m \in \mathbb{Z}_g)$: Generate a random β-bit integer vector $\mathbf{b} = (b_1, \ldots, b_\tau)^t$ and output

$$c = \Big(m + g \cdot \sum_{i=1}^{\tau} b_i \cdot x_i\Big) \bmod x_0 \,.$$

$\mathsf{DGHV.Add}(\mathsf{pk}, c_1, c_2)$: Output $c \leftarrow (c_1 + c_2) \bmod x_0$.

$\mathsf{DGHV.Mult}(\mathsf{pk}, c_1, c_2)$: Output $c \leftarrow (c_1 \cdot c_2) \bmod x_0$.

$\mathsf{DGHV.Decrypt}(\mathsf{sk}, c)$: Output $m \leftarrow (c \bmod p) \bmod g$.

For a large enough η, this scheme is somewhat homomorphic, *i.e.* a limited number of homomorphic operations can be performed on ciphertexts. More precisely, define $\mathsf{noise}_p(c) = |c \bmod p|$ for any integer c. Given two ciphertexts c_1 and c_2 of $m_1 \in \mathbb{Z}_g$ and $m_2 \in \mathbb{Z}_g$ where $\mathsf{noise}_p(c_1)$ is a ρ_1-bit integer and $\mathsf{noise}_p(c_2)$ a ρ_2-bit integer, the ciphertext $(c_1 + c_2 \bmod x_0)$ is an encryption of $(m_1 + m_2 \bmod g)$ with noise bit-length smaller than $\max(\rho_1, \rho_2) + 1$, and the ciphertext $(c_1 \cdot c_2 \bmod x_0)$ is an encryption of $(m_1 \cdot m_2 \bmod g)$ with noise bit-length smaller than $\rho_1 + \rho_2 + \alpha$.

In particular, for a freshly generated ciphertext c, we have $\log_2 \mathsf{noise}(c) < \beta + \alpha + \rho + \log_2 \tau$, and since the ciphertext noise must remain smaller than p to maintain correctness, the scheme roughly allows $\eta/(\beta + \alpha + \rho + \log_2 \tau)$ sequential multiplications on ciphertexts.

Correctness. Let us prove the correctness of the scheme. We recall the definition from [Gen09, vDGHV10, CMNT11]. Consider a homomorphic public-key encryption scheme $\mathcal{E}$ with an additional algorithm Eval taking as input the public key pk, a mod-g arithmetic circuit C with t inputs and t ciphertexts c_i, and outputting another ciphertext c.

Definition 7.15 (Correct Homomorphic Decryption)**.** The scheme $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Eval})$ is correct for a given t-input circuit C if, for any key-pair $(\mathsf{sk}, \mathsf{pk})$ output by $\mathsf{Keygen}(\lambda)$, any t plaintext bits $m_1, \ldots, m_t$, and any ciphertexts $\mathbf{c} = (c_1, \ldots, c_t)^t$ with $c_i \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_i)$, it holds that $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, C, \mathbf{c})) = C(m_1, \ldots, m_t)$.

As in [Gen09, vDGHV10, CMNT11], we define a *permitted circuit* as one where for any $i \geqslant 1$ and any set of integers inputs less that $\tau^i \cdot g^i \cdot 2^{i \cdot (\rho+\beta)}$ in absolute value, the generalized circuit's output has absolute value at most $2^{i(\eta-3-n)}$ with $n = \lceil \log_2(\lambda + 1) \rceil$; we let $\mathcal{C}_{\mathsf{DGHV}}$ be the set of permitted circuits for the DGHV scheme described above. We have

Lemma 7.16. *The scheme from above is correct for $\mathcal{C}_{DGHV}$.*

Proof. The proof is essentially similar to the proof in [CMNT11, Appendix B] – we detail it for completeness. Given a ciphertext c outputted by $\mathsf{DGHV.Encrypt}(\mathsf{pk}, m)$, there exists an integer vector $\mathbf{b} = (b_i)_{1 \leqslant i \leqslant \tau} \in \{0,1\}^\tau$ such that

$$c = m + g \cdot \sum_{i=1}^{\tau} b_i \cdot x_i \bmod x_0$$

This gives

$$\mathsf{noise}(c) = |c \bmod p| \leqslant g \cdot \tau \cdot 2^{\rho+\beta} \,.$$

Let $C \in \mathcal{C}_{\mathsf{DGHV}}$ be a permitted circuit with t inputs and let $C^\dagger$ be the corresponding circuit operating over the integers rather than modulo g. Let $c_i \leftarrow \mathsf{DGHV.Encrypt}(\mathsf{pk}, m_i)$. Define $c = C^\dagger(c_1, \ldots, c_t)$. We have

$$c \bmod p = C^\dagger(c_1, \ldots, c_t) \bmod p = C^\dagger(c_1 \bmod p, \ldots, c_t \bmod p) \bmod p \,.$$

From the definition of permitted circuits, we obtain

$$|C^\dagger(c_1 \bmod p, \ldots, c_t \bmod p)| \leqslant 2^{\eta-4} \leqslant p/8\,.$$

Therefore $C^\dagger(c_1 \bmod p, \ldots, c_t \bmod p) \bmod p = C^\dagger(c_1 \bmod p, \ldots, c_t \bmod p)$ and then

$$c \bmod p = C^\dagger(c_1 \bmod p, \ldots, c_t \bmod p)\,.$$

Finally,

$$\begin{aligned}(c \bmod p) \bmod g &= C^\dagger(c_1 \bmod p, \ldots, c_t \bmod p) \bmod g \\ &= C^\dagger\big((c_1 \bmod p) \bmod g, \ldots, (c_t \bmod p) \bmod g\big) \\ &= C(m_1, \ldots, m_t)\,.\end{aligned}$$

□

Semantic Security. Let us prove that this scheme is semantically secure under the assumption that the decisional EF-AGCD problem is hard. More precisely we have the following theorem:

Theorem 7.17. *The above one-slot DGHV scheme is semantically secure under the $(\gamma, \eta, \nu, \rho)$ decisional Error-Free Approximate-GCD assumption, assuming that $\beta \cdot \tau > \gamma + 2\lambda$.*

To prove the theorem, we use a preliminary lemma from [KLYC13] stating that the distribution of the public-key elements is indistinguishable from random elements in $[0, x_0)$ if the decisional EF-AGCD problem is hard.

Lemma 7.18. *For the parameters $(\gamma, \eta, \nu, \rho)$, let $\mathsf{pk} = \{x_i\}_{i=0}^{\tau}$ and $\mathsf{sk} = p$ be chosen as in the* Keygen *procedure. Define $\mathsf{pk}' = \{x_0, x'_1, \ldots, x'_\tau\}$ for x'_i uniformly generated in $[0, x_0)$. Then pk and pk' are indistinguishable under the decisional EF-AGCD assumption.*

Proof. Assume that there exists a polynomial-time distinguisher $\mathcal{B}$ distinguishing pk from pk' with advantage ε. Using $\mathcal{B}$ we can construct a polynomial-time distinguisher $\mathcal{A}$ solving the decisional EF-AGCD problem with advantage ε/τ. For $r = 0, \ldots, \tau$, define

$$\mathsf{pk}^{(r)} = \{x_0, x_1^{(r)}, \ldots, x_r^{(r)}, x_{r+1}^{(r)}, \ldots, x_\tau^{(r)}\}\,,$$

where $x_1^{(r)}, \ldots, x_r^{(r)} \leftarrow [0, x_0)$ and $x_{r+1}^{(r)}, \ldots, x_\tau^{(r)} \leftarrow \mathcal{D}_\rho(p, q_0)$. Thus $\mathcal{B}$ has advantage ε to distinguish $\mathsf{pk}^{(0)} = \mathsf{pk}$ from $\mathsf{pk}^{(\tau)} = \mathsf{pk}'$. By a standard hybrid argument, there exists a r so that $\mathcal{B}$ distinguish $\mathsf{pk}^{(r)}$ and $\mathsf{pk}^{(r+1)}$ with advantage ε/τ; therefore letting $x_r^{(r)} = c$ where c is the decisional EF-AGCD challenge, $\mathcal{B}$ allows us to solve the decisional EF-AGCD problem with advantage ε/τ. □

Proof of Theorem 7.17. Under the attack scenario the attacker first receives the public key, then transmits two messages $m_0, m_1 \in \mathbb{Z}_g$, and receives an encryption of m_b. The attacker outputs a guess b' and succeeds if $b' = b$. We use a sequence of games and denote by S_i the event that the attacker succeeds in $\mathbf{Game}_i$.

$\mathbf{Game}_0$: This is the attack scenario. We simulate the challenger by running Keygen to obtain pk and sk.

$\mathbf{Game}_1$: We replace the x_i's in the public key by elements uniformly drawn in $[0, x_0)$. By Lemma 7.18, we have

$$|\Pr[S_1] - \Pr[S_0]| \leqslant \tau \cdot \varepsilon_{\text{decisional EF-AGCD}}\,.$$

$\mathbf{Game}_2$: Since x_0 is 2^ν-rough and $\nu > \lambda$, we get the result of the Corollary 3.4 of the Leftover Hash Lemma (*cf.* Section 3.3.1) is valid for $q = x_0$, for an overwhelming proportion of the x_i's. Therefore, this yields $\sum_{i=1}^{\tau} b_i \cdot x_i \bmod x_0$ is ε-statistically indistinguishable from uniform modulo x_0, with $\varepsilon = 2^{(\gamma-\beta\cdot\tau)/2}$. This advantage is negligible since $\beta \cdot \tau > \gamma + 2\lambda$. Therefore we can replace the challenge ciphertext by a uniform integer modulo x_0; this no longer gives any information on m_b and therefore $\Pr[S_2] = 1/2$. Moreover we have $|\Pr[S_2] - \Pr[S_1]| \leqslant \varepsilon$, which concludes the proof. □

Remark 7.19. In the initial works on the DGHV scheme [vDGHV10, CMNT11, CNT12], the security of DGHV was based on the computational EF-AGCD problem. This requires an additional term $g \cdot r'$ during encryption where $\log_2(r') > \rho + \beta + \log_2 \tau + \lambda$ to drown the noise in the ciphertext (because we cannot apply the Leftover Hash Lemma anymore). This yields a noticeable loss in performance and noticeable growth in ciphertext size, while no attack is known to be applicable to the decisional-EF-AGCD and not to the computational-EF-AGCD – *cf.* Section 7.2.

7.3.2 Multi-Slot DGHV Scheme

In this section, we describe a multi-slot variant of DGHV. The key idea is to use the Chinese Remainder Theorem to encrypt elements of $\mathcal{M} = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_\ell}$ where the g_i's are integers (such that for all i, $g_i < 2^\alpha$), into a single ciphertext. The plaintext space $\mathcal{M}$ is a $\mathbb{Z}$-module; denote $\mathbf{e}_1, \ldots, \mathbf{e}_\ell$ its canonical basis, *i.e.* the basis such that $\mathbf{e}_i[j] = 1$ when $i = j$ and 0 otherwise. Thanks to the structure of the ciphertexts, homomorphic additions and multiplications will be applied in parallel and componentwise over each coordinate while performing integer additions and multiplications.

Key Idea. To encrypt a vector $\mathbf{m} = (m_1, \ldots, m_\ell)^t \in \mathcal{M}$ into a single ciphertext c, we use the Chinese Remainder Theorem with respect to a tuple of $n+1$ pairwise coprime integers $q_0, p_1, \ldots, p_\ell$. Define $x_0 = q_0 \cdot \prod_{i=1}^{\ell} p_i$. The ciphertext has the form

$$c = \mathsf{CRT}_{q_0,p_1,\ldots,p_\ell}(q, g_1 \cdot r_1 + m_1, \ldots, g_\ell \cdot r_\ell + m_\ell)\,, \tag{7.11}$$

where q is randomly chosen modulo q_0. It correctly decrypts to $\mathbf{m}$ by computing $m_i = (c \bmod p_i) \bmod g_i$ for all $1 \leqslant i \leqslant \ell$. Note that we can write $c = q_i \cdot p_i + g_i \cdot r_i + m_i$; thus modulo p_i, the ciphertext c behaves as in the one-slot variant. Therefore, the addition (resp. multiplication) of two ciphertexts yields a new ciphertext that decrypts, on slot i, to the componentwise sum (resp. product) modulo g_i of the original plaintexts.

Multi-Slot Scheme. Let us present the public key multi-slot DGHV scheme (BDGHV):

BDGHV.Keygen$(1^\lambda, g_1, \ldots, g_\ell)$: Generate ℓ 2^ν-rough η-bit integer p_i's and randomly generate a 2^ν-rough integer $q_0 \in [0, 2^\gamma/\pi)$ where $\pi = \prod_i p_i$. Denote $x_0 = \pi \cdot q_0$.

Next for $1 \leqslant i \leqslant \tau$, define x_i the encryption of $\mathbf{0} = (0, \ldots, 0)^t \in \mathcal{M}$ following Equation (7.11) with a uniform q modulo q_0, and uniform r_i's over $[0, 2^\rho)$.

Then for $1 \leqslant i \leqslant \ell$, define x'_i the encryption of $\mathbf{e}_i \in \mathcal{M}$ following Equation (7.11) with a uniform q modulo q_0, and uniform r_i's over $[0, 2^\rho)$.

Let $\mathsf{sk} = \{p_1, \ldots, p_\ell\}$ and $\mathsf{pk} = \{x_0, x_1, \ldots, x_\tau, x'_1, \ldots, x'_\ell\}$.

BDGHV.Encrypt$(\mathsf{pk}, \mathbf{m} \in \mathcal{M})$: Generate a random β-bit integer vector $\mathbf{b} = (b_1, \ldots, b_\tau)^t$ and output

$$c = \Big(\sum_{i=1}^{\ell} m_i \cdot x'_i + \sum_{i=1}^{\tau} b_i \cdot x_i \Big) \bmod x_0\,.$$

BDGHV.Add(pk, c_1, c_2): Output $c \leftarrow (c_1 + c_2) \bmod x_0$.

BDGHV.Mult(pk, c_1, c_2): Output $c \leftarrow (c_1 \cdot c_2) \bmod x_0$.

BDGHV.Decrypt(sk, c): Output $\mathbf{m} = (m_1, \ldots, m_\ell)^t$ where $m_i \leftarrow (c \bmod p_i) \bmod g_i$ for all i.

As previously this scheme is somewhat homomorphic: if $\lceil \log_2 \max(g_i) \rceil = \alpha$, the scheme roughly allows $\eta/(\beta + \alpha + \rho + \log_2 \tau + 1)$ successive multiplications on ciphertexts.

Remark 7.20. The main noticeable difference with the one-slot variant is the public key elements x'_i's which are designed so that the m_i's are disposed at the right place in the ciphertext. However, this can be seen as a natural extension of the one-slot variant: we could have published an encryption x' of the value 1 and encrypt by computing $c \leftarrow (m \cdot x' + g \cdot \sum_i b_i \cdot x_i) \bmod x_0$. Now $x' = 1$ yields the same result and reduces the public-key size and the noise in the fresh ciphertexts; this is therefore the wisest choice to make.

Correctness. We adapt to the batch settings the definition of correctness from [Gen09, vDGHV10] (see also Definition 7.15). We consider an homomorphic public-key encryption scheme $\mathcal{E}$ with plaintext space $\mathcal{M}$, an additional algorithm Eval taking as input the public key pk, an arithmetic circuit C over the integers with t inputs and t ciphertexts c_i, and outputting another ciphertext c.

Definition 7.21 (Correct batch homomorphic decryption)**.** The scheme $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Eval})$ of plaintext space $\mathcal{M} = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_\ell}$ is correct for a given t-input circuit C if, for any key-pair $(\mathsf{sk}, \mathsf{pk})$ output by $\mathsf{Keygen}(\lambda)$, any t plaintext ℓ-bit vectors $\mathbf{m}_1, \ldots, \mathbf{m}_t$, and any ciphertexts $\mathbf{C} = (c_1, \ldots, c_t)^t$ with $c_i \leftarrow \mathsf{Encrypt}(\mathsf{pk}, \mathbf{m}_i)$, it holds that

$$\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, C, \mathbf{C})) = \Big(C_1(\mathbf{m}_1[1], \ldots, \mathbf{m}_t[1]), \ldots, C_\ell(\mathbf{m}_1[\ell], \ldots, \mathbf{m}_t[\ell])\Big)\,,$$

where C_i is the circuit C with circuit operations modulo g_i rather than over the integers.

As in [Gen09, vDGHV10], we define a *permitted circuit* as one where for any $i \geqslant 1$ and any set of integers inputs less that $\tau^i 2^{i(\alpha+\rho+\beta+1)}$ in absolute value, the generalized circuit's output has absolute value at most $2^{i(\eta-3-n)}$ with $n = \lceil \log_2(\lambda+1) \rceil$; we let $\mathcal{C}_{\mathsf{BDGHV}}$ be the set of permitted circuits.

We have the following result:

Lemma 7.22. *The BDGHV encryption scheme is correct for $\mathcal{C}_{BDGHV}$.*

Proof. Given a ciphertext c outputted by $\mathsf{BDGHV.Encrypt}(\mathsf{pk}, \mathbf{m})$, there exist an integer vectors $\mathbf{b} = (b_i)_{1 \leqslant i \leqslant \tau} \in [0, 2^\beta)^\tau$ such that

$$c = \sum_{i=1}^{\ell} m_i \cdot x'_i + \sum_{i=1}^{\tau} b_i \cdot x_i \bmod x_0\,.$$

For each $j = 1, \ldots, \ell$, this gives

$$|c \bmod p_j| \leqslant \ell \cdot 2^{\rho+\alpha} + \tau \cdot 2^{\alpha+\rho+\beta} \leqslant \tau \cdot 2^{\alpha+\rho+\beta+1}. \tag{7.12}$$

Let C be a permitted circuit with t inputs. Let $c_i \leftarrow \mathsf{BDGHV.Encrypt}(\mathsf{pk}, \mathbf{m}_i)$. We have, for each $j = 1, \ldots, \ell$,

$$c \bmod p_j = C(c_1, \ldots, c_t) \bmod p_j = C(c_1 \bmod p_j, \ldots, c_t \bmod p_j) \bmod p_j. \tag{7.13}$$

From (7.12) and the definition of permitted circuits, we obtain

$$|C(c_1 \bmod p_j, \ldots, c_t \bmod p_j)| \leqslant 2^{\eta-4} \leqslant p_j/8\,.$$

Therefore, from (7.13), we get that $c \bmod p_j = C(c_1 \bmod p_j, \ldots, c_t \bmod p_j)$, and eventually

$$[c \bmod p_j]_{g_j} = \big[C([c_1 \bmod p_j]_{g_j}, \ldots, [c_t \bmod p_j]_{g_j})\big]_{g_j} = C_j(\mathbf{m}_1[j], \ldots, \mathbf{m}_t[j])\,,$$

which concludes the proof. □

Semantic Security. Let us prove our multi-slot DGHV scheme is semantically secure under the same assumption as the one-slot scheme, *i.e.* the decisional Error-Free Approximate-GCD assumption from Definition 7.4.

Theorem 7.23. *The above multi-slot DGHV scheme is semantically secure under the $(\gamma - (\ell-1)\eta, \eta, \nu, \rho)$ decisional Error-Free Approximate-GCD assumption, assuming that $\beta \cdot \tau > \gamma + 2\lambda$.*

As for the one-slot semantic security proof, we use a preliminary lemma from [KLYC13] stating that the distribution of the public-key elements is indistinguishable from random elements in $[0, x_0)$ if the decisional ℓ-EF-AGCD problem is hard.

Lemma 7.24. *For the parameters $(\gamma, \eta, \nu, \rho)$ and ℓ, let $\mathsf{pk} = \{x_0, x'_1, \ldots, x'_\ell, x_1, \ldots, x_\tau\}$ and $\mathsf{sk} = \{p_1, \ldots, p_\ell\}$ be chosen as in the BDGHV.*Keygen *procedure. Define $\mathsf{pk}' = \{x_0, x'_1, \ldots, x'_\ell, x''_1, \ldots, x''_\tau\}$ for x'_i uniformly generated in $[0, x_0)$. Then pk and pk' are indistinguishable under the decisional ℓ-EF-AGCD assumption.*

Proof. The proof is identical to the proof of Lemma 7.18: by a standard hybrid argument, we can show that any polynomial-time distinguisher, having advantage ε, can be turned into a polynomial-time distinguisher solving the decisional ℓ-EF-AGCD problem with advantage ε/τ. □

Proof of Theorem 7.23. Using the same proof than for the one-slot variant, *i.e.* of Theorem 7.17, we can show that the multi-slot DGHV scheme is secure under the $(\gamma, \eta, \nu, \rho)$ decisional ℓ-Error-Free Approximate-GCD assumption. Now, by Corollary 7.12, this latter problem is reducible to the $(\gamma - (\ell - 1)\eta, \eta, \nu, \rho)$ decisional Error-Free Approximate-GCD assumption, which concludes the proof. □

Remark 7.25. In the full version of the article [CLT13a] cosigned with J.-S. Coron and M. Tibouchi, the security of the BDGHV scheme was based on the computational EF-AGCD problem. In particular, we described a method to independently randomize a ciphertext c modulo each of the p_i's, without knowing the p_i's. The encryption procedure of the resulting scheme makes use of another subset sum of public key elements which, taken modulo each of the p_i's, generate a lattice with special properties.

This method later became the key idea behind our multilinear maps candidate over the integers [CLT13b] (also cosigned with J.-S. Coron and M. Tibouchi); more precisely, we introduce a Leftover-Hash Lemma over lattices (*cf.* Section 11.4).

7.3.3 Asymptotic Parameters

In order to select parameters for a target level of security λ, the constraints given in Table 7.2 must be verified.

Table 7.2 – Asymptotic Constraints on DGHV and BDGHV Parameters.

Scheme	DGHV	BDGHV	Reason
$\rho \geqslant$	2λ		to avoid brute force attack on the noise [CN12]
$\eta \geqslant$	$\rho + \alpha + \beta + \log_2(\tau)$	$\rho + \alpha + \beta + \log_2(\tau) + 1$	for correct decryption
$\eta =$	$\rho \cdot \Omega(\lambda \log^2 \lambda)$		for homomorphically evaluating the "squashed decryption" circuit (*cf.* [vDGHV10, CMNT11] and Section 7.4)
$\gamma \geqslant$	$\eta^2 \cdot \Omega(\lambda)$		in order to thwart lattice-based attacks [vDGHV10, CMNT11]
$\beta \cdot \tau \geqslant$	$\gamma + 2\lambda$		in order to apply the Leftover Hash Lemma

To satisfy the constraints of Table 7.2, one can take

$$\rho = 2\lambda, \quad \eta = \tilde{\mathcal{O}}(\lambda^2), \quad \gamma = \tilde{\mathcal{O}}(\lambda^5), \quad \beta = \tilde{\mathcal{O}}(\lambda^2), \quad \tau = \tilde{\mathcal{O}}(\lambda^3)$$

as in [CNT12], and $\ell = \tilde{\mathcal{O}}(\lambda^2)$. The main difference between the BDGHV scheme and the DGHV scheme is that, in the former, the ciphertext expansion ratio becomes $\gamma/\ell = \tilde{\mathcal{O}}(\lambda^3)$ instead of $\gamma = \tilde{\mathcal{O}}(\lambda^5)$. However, the public key size using the compressed public key technique from [CNT12] (cf Section 7.5) becomes $\tilde{\mathcal{O}}(\lambda^7)$ instead of $\tilde{\mathcal{O}}(\lambda^5)$. We refer to Section 7.6 for concrete parameters and timings.

7.3.4 Advantages of the Multi-Slot Variant

The multi-slot variant is not a mere useless generalization of the one-slot variant; in particular, a certain number of problems arising in the one-slot variant can be solved by this multi-slot variant. Indeed, let us denote $\alpha = \log_2 g$. To multiply m ciphertexts in the first variant, one has to select η such that

$$\eta > m \cdot (\alpha + \rho)$$

In particular, if one wants to work with integers of 64 bits and to perform 10 multiplications modulo $g = 2^{64}$, η has to verify $\eta > 640 + 800 \approx 1500$ for 80 bits of security, and this will yield $\gamma \approx 10000000$. Now if one wants to multiply 10 64-bit numbers over $\mathbb{Z}$, one must select $g = 2^{640}$ and $\eta \approx 7500$, which is quite impractical.

Now, for the multi-slot variants, one can select $\log_2 g_i = \alpha$ with coprime g_i's and the message space will be isomorphic to $\mathbb{Z}/g\mathbb{Z}$ where $g = \prod g_i$ is at least a $\ell \cdot (\alpha - 1)$ bit numbers. In particular, allowing a 10% increase in the ciphertext size, *i.e.* $\gamma = 11000000$, one can still set $\alpha = 64$ and $\eta = 1500$ (so that one can do 10 multiplications), but $\ell = 666$, and one can homomorphically multiply integers over $\mathbb{Z}$ as long as the result is smaller than 2^{52614}. This extreme example shows that a useful trade-off could yield very interesting results as far as homomorphic operations over *integers* are considered.

In other words, the multi-slot variant exploits the fact that the size of the ciphertext γ is substantially larger than the size of the secret key η to consider several slots at a small additional cost. As a consequence, it yields a huge parallelization capability, or increases exponentially the size of the plaintext space.

7.4 Making the Scheme Fully Homomorphic

In this section, we follow Gentry's blueprint [Gen09] to transform a somewhat homomorphic encryption scheme into a fully homomorphic encryption scheme. This was described for the one-slot DGHV scheme over the integers in [vDGHV10, CMNT11], therefore we describe it for the batch DGHV scheme, *i.e.* the multi-slot DGHV scheme described in Section 7.3.2.

Throughout the whole section, we assume that $g_i = 2$ for all i, *i.e.* $\mathcal{M} = \mathbb{Z}_2^\ell$ is the set of ℓ-bit vectors. Indeed, that is the only setting for which we know how to transform the somewhat homomorphic encryption scheme of Section 7.3 into a *fully* homomorphic encryption scheme.

7.4.1 The Squashed Scheme

In order to follow Gentry's blueprint and make our somewhat homomorphic scheme amenable to bootstrapping, we first need to squash the decryption circuit, *i.e.* change the decryption procedure so as to express it as a low degree polynomial in the bits of the secret key.

We use the same technique as in the original DGHV scheme [vDGHV10], but generalize it to the batch setting. Three more parameters are added: $\kappa = \gamma + 2 + \lceil \log_2(\theta + 1) \rceil$, $\theta = \lambda$ and $\Theta = \tilde{\mathcal{O}}(\lambda^3)$. These parameters are selected as in [CMNT11] to ensure that solving the sparse subset sum is intractable.

We add to the public key a set $\mathbf{y} = \{y_1, \ldots, y_\Theta\}$ of rational numbers in $[0, 2)$ with κ bits of precision after the binary point, such that for all $1 \leqslant j \leqslant \ell$ there exists a sparse subset $S_j \subset [1, \Theta]$ of size θ with $\sum_{i \in S_j} y_i \simeq 1/p_j \bmod 2$. The secret-key is replaced by the indicator vector of the subsets S_j. Formally the scheme is modified as follows:

BDGHV.Keygen(1^λ). Generate $\mathsf{sk}^* = (p_1, \ldots, p_\ell)$ and pk^* as in Section 7.3.2. Set $x_{p_j} \leftarrow \lfloor 2^\kappa / p_j \rceil$ for $j = 1, \ldots, \ell$. Choose at random Θ-bit vectors $\mathbf{s}_j = (s_{j,1}, \ldots, s_{j,\Theta})^t$, each of Hamming weight θ, for $1 \leqslant j \leqslant \ell$. Choose at random Θ integers $u_i \in [0, 2^{\kappa+1})$ for $1 \leqslant i \leqslant \Theta$, fulfilling the condition that $x_{p_j} = \sum_{i=1}^{\Theta} s_{j,i} \cdot u_i \bmod 2^{\kappa+1}$ for all j. Set $y_i = u_i / 2^\kappa$ and $\mathbf{y} = (y_1, \ldots, y_\Theta)^t$. Hence, each y_i is a positive number smaller than two, with κ bits of precision after the binary point, and verifies

$$\frac{1}{p_j} = \sum_{i=1}^{\Theta} s_{j,i} \cdot y_i + \varepsilon_j \bmod 2 \tag{7.14}$$

for some $|\varepsilon_j| < 2^{-\kappa}$.

Output the secret key $\mathsf{sk} = (\mathbf{s}_1, \ldots, \mathbf{s}_\ell)$ and public key $\mathsf{pk} = (\mathsf{pk}^*, y_1, \ldots, y_\Theta)$.

$\mathsf{BDGHV.Expand}(\mathsf{pk}, c)$. The ciphertext expansion procedure takes as input a ciphertext c and computes an expanded ciphertext: for every $1 \leqslant i \leqslant \Theta$, compute z_i given by $z_i = \lfloor c \cdot y_i \rceil \bmod 2$ with $n = \lceil \log_2(\theta + 1) \rceil$ bits of precision after the binary point. Define the vector $\mathbf{z} = (z_i)_{i=1,\ldots,\Theta}$ and output the expanded ciphertext $(c, \mathbf{z})$.

$\mathsf{BDGHV.Decrypt}(\mathsf{sk}, c, \mathbf{z})$. Output $\mathbf{m} = (m_1, \ldots, m_\ell)$ with

$$m_j \leftarrow \left[\left\lfloor \sum_{i=1}^{\Theta} s_{j,i} \cdot z_i \right\rceil \right]_2 \oplus (c \bmod 2). \tag{7.15}$$

This completes the description of the scheme. We use $n = \lceil \log_2(\theta + 1) \rceil$ as in [CMNT11].

The definition of a permitted circuit does not seem to give an easy criterion to determine whether a given computation is permitted. As in [CMNT11, vDGHV10], we provide a sufficient condition for a multivariate polynomial f for a circuit C to be permitted. If f is of degree d and the sum of the absolute values of its coefficients if $\|f\|_1$, then $C \in \mathcal{C}_{\mathsf{BDGHV}}$ provided that:

$$d \leqslant \frac{\eta - 3 - n - \log_2 \|f\|_1}{\beta + \rho + \alpha + 1 + \log_2 \tau}.$$

Following [vDGHV10], we refer to such polynomials f as *permitted polynomials*, and denote the set of these polynomials by $\mathcal{P}_{\mathsf{BDGHV}}$.

The proof of the following Lemma is the same as in [CMNT11, Appendix E].

Lemma 7.26. *The BDGHV encryption scheme is correct for the set $\mathcal{C}(\mathcal{P}_\mathcal{E})$ of circuits that compute permitted polynomials.*

Remark 7.27. To reduce the size of the public key we can generate all the y_i's pseudo-randomly as in [CMNT11], except ℓ of them in order to satisfy Equation (7.14) for all $1 \leqslant j \leqslant \ell$.

7.4.2 Bootstrapping

As in [vDGHV10], we get that the BDGHV scheme is bootstrappable. Moreover, the Recrypt procedures works naturally in *parallel* over the plaintext bits.

In the original DGHV scheme [vDGHV10, CMNT11], the decryption equation was:

$$m \leftarrow \left[\left\lfloor \sum_{i=1}^{\Theta} s_i \cdot z_i \right\rceil \right]_2 \oplus (c \bmod 2) \tag{7.16}$$

and could be homomorphically evaluated by providing an encryption σ_i of every secret-key bit s_i; one would obtain a new ciphertext which would encrypt the same plaintext bit m but with a possibly reduced noise.

Similarly, the decryption Equation (7.15) for the batch scheme can be evaluated homomorphically by providing for all $1 \leqslant i \leqslant \Theta$ an encryption σ_i of the ℓ secret-key bits $s_{j,i}$, with:

$$\sigma_i = \mathsf{BDGHV.Encrypt}(s_{1,i}, \ldots, s_{\ell,i}).$$

This gives a new ciphertext that encrypts the same ℓ-bit plaintext vector, but with a (possibly) reduced noise. In other words, instead of having an homomorphic evaluation of a single Equation (7.16), we have that the ℓ equations in (7.15) are homomorphically evaluated in parallel, one in each of the ℓ plaintext slots of the ciphertext. Therefore the Recrypt operation is done in parallel over the ℓ slots, with the same complexity as a single Recrypt operation in the original scheme.

From Gentry's theorem, we obtain a homomorphic encryption scheme for circuits of any depth. The proof of the following theorem is identical to the proof of Theorem 5.1 in [CMNT11].

Theorem 7.28. *Let D_{BDGHV} be the set of augmented (squashed) decryption circuits for the BDGHV scheme. Then $D_{\mathsf{BDGHV}} \subset C(\mathcal{P}_{\mathsf{BDGHV}})$.*

7.4.3 Complete Set of Operations for Plaintext Vectors

From what precedes, we can implement homomorphic SIMD-type operations on our packed ciphertexts, where the Add and Mult operations are applied to ℓ different input bits at once. However, a desired feature when dealing with packed ciphertexts is the ability to move values between plaintext slots with a public Permute operation. As opposed to [GHS12b] we cannot rely on an underlying algebraic structure. Instead we show how to perform such Permute at ciphertext refresh time. This feature is therefore supported at no extra cost assuming a ciphertext refresh operation has to be carried out anyway (*i.e.* after each Mult gate). Note that a similar technique was described independently in [BGH13] for the RLWE-based fully homomorphic schemes [BV11a, BV11b, GHS12b].

For any permutation ζ over $\{1, \ldots, \ell\}$, we want to homomorphically evaluate the function

$$\ell\text{-Permute}\,(\zeta, (u_1, \ldots, u_\ell)) = \left(u_{\zeta(1)}, \ldots, u_{\zeta(\ell)}\right).$$

Let ζ be a permutation to be applied homomorphically on the plaintext bits. During the Keygen operation, one can define for each $i \in [1, \Theta]$

$$\sigma_i^\zeta = \mathsf{BDGHV.Encrypt}(s_{\zeta(1),i}, \ldots, s_{\zeta(\ell),i}).$$

Now, performing the ciphertext refresh operation ("recryption") with the σ_i^ζ's instead of the σ_i's gives a ciphertext of the plaintext vector $(m_{\zeta(1)}, \ldots, m_{\zeta(\ell)})$ which is exactly the desired result. Therefore any permutation ζ can be implemented by putting the corresponding σ_i^ζ's in the public key.

To be able to perform arbitrary permutations on the plaintext vector, one can augment the public key by a minimal set of permutations ζ's that generates the whole permutation group $\mathfrak{S}_\ell$, such as the transposition $(1, 2)$ and the cycle $(1, 2, \ldots, \ell)$. In that case the impact on the public key is small (as only $2 \cdot \Theta \cdot \gamma$ bits are added), but the performance overhead is significant, since as many as $\mathcal{O}(\ell)$ ciphertext refresh operations may be needed to carry out a desired permutation.

A more practical solution is to use a Beneš network [Ben64] of permutations as in [GHS12b]. In that case it suffices to add $2\log_2(\ell)$ permuting elements to the public key to enable circular rotations by $\pm 2^i$ bit position. Then any permutation can be obtained in $(2\log(\ell) - 1)$ steps. At each step, at most two rotations and two Select operations are performed, where the Select operation on c_1 and c_2 constructs a ciphertext where each of the ℓ plaintext slot is chosen either from c_1 or c_2; such Select operation is easily obtained with two Mult (and two recryptions) and one Add, see [GHS12b]. This approach has a limited impact on the public key ($2\log_2(\ell) \cdot \Theta \cdot \gamma$ more bits), and any permutation can then be performed with at most $6 \cdot (2\log_2 \ell - 1)$ recryptions.

In practice, however, the circuit to be homomorphically evaluated is likely to be known in advance, so it is possible to put a set of distinguished permutations in the public key that provides an optimal time-memory trade-off. For example, in Chapter 10, we describe two variants of homomorphic evaluations of the full AES circuit that require respectively only *four* permutations and no permutation at all.

7.5 Complete Description of the Batch DGHV Scheme with Compressed Public Keys

In this section, we provide a complete description of our multi-slot FHE scheme with the ciphertext compression technique of [CNT12]. Note that, as in [CNT12], the ciphertext compression technique is applied to both the public key elements x_i's and x'_i's of the somewhat homomorphic scheme, and to the encryptions σ_i's of the secret key bits. The ciphertext compression technique enables to compress a ciphertext from $\gamma = \tilde{\mathcal{O}}(\lambda^5)$ bits down to $\ell \cdot \eta + \lambda = \ell \cdot \tilde{\mathcal{O}}(\lambda^2)$ bits.

7.5.1 Description

$\mathsf{BDGHV.Keygen}(1^\lambda)$. Generate ℓ 2^ν-rough η-bit integer p_i's and randomly generate a 2^ν-rough integer $q_0 \in [0, 2^\gamma/\pi)$ where $\pi = \prod_i p_i$. Denote $x_0 = \pi \cdot q_0$.

Initialize a pseudo-random generator f_1 with a random seed se_1. Use $f_1(\mathsf{se}_1)$ to generate a set of integers $\chi_i \in [0, x_0)$ for $i \in [1, \tau]$ and $\chi'_i \in [0, x_0)$ for $i \in [1, \ell]$ for $i \in [0, \ell - 1]$.

Define γ-bit integers as follows:

1. the integers x_i's $(1 \leqslant i \leqslant \tau)$ such that $x_i = \chi_i - \Delta_i$ with
$$\Delta_i = [\chi_i]_\pi + \xi_i \cdot \pi - \mathsf{CRT}_{p_1,\ldots,p_\ell}(r_{i,1}, \ldots, r_{i,\ell})$$
where $r_{i,j} \leftarrow \mathbb{Z} \cap [0, 2^\rho)$ and $\xi_i \leftarrow \mathbb{Z} \cap [0, 2^{\lambda+\ell\cdot\eta}/\pi)$;
2. the integers x'_i's $(1 \leqslant i \leqslant \ell)$ such that $x'_i = \chi'_i - \Delta'_i$ with
$$\Delta'_i = [\chi'_i]_\pi + \xi'_i \cdot \pi - \mathsf{CRT}_{p_1,\ldots,p_\ell}(2r'_{i,1} + \delta_{i,1}, \ldots, 2r'_{i,\ell} + \delta_{i,\ell})$$
where $r'_{i,j} \leftarrow \mathbb{Z} \cap [0, 2^\rho)$ and $\xi'_i \leftarrow \mathbb{Z} \cap [0, 2^{\lambda+\ell\cdot\eta}/\pi)$.

Additionally, generate at random Θ-bit vectors $\mathbf{s}_j = (s_{j,1}, \ldots, s_{j,\Theta})^t$ for $1 \leqslant j \leqslant \ell$, each split in θ boxes of size $B = \Theta/\theta$ each, with exactly one non-zero bit in each box, and such that $s_{j,j} = 1$ for each $j = 1, \ldots, \ell$. Initialize a pseudo-random generator f_2 with a random seed se_2, and use $f_2(\mathsf{se}_2)$ to generate integers $u_i \in [0, 2^{\kappa+1})$ for $i \in \mathbb{Z} \cap [1, \Theta]$. Then set $u_1, \ldots, u_\ell$ such that
$$x_{p_j} = \sum_{i=1}^{\Theta} s_{j,i} \cdot u_i \bmod 2^{\kappa+1},$$
where $x_{p_j} = \lfloor 2^\kappa / p_j \rceil$ for each $j = 1, \ldots, \ell$.

Initialize a pseudo-random generator f_3 with a random seed se_3. Use $f_3(\mathsf{se}_3)$ to generate a set of integers $\chi^\sigma_i \in [0, x_0)$ for $i \in [1, \Theta]$.

For $i \in [1, \Theta]$, define the γ-bit integers $\sigma_i = \chi^\sigma_i - \Delta^\sigma_i$ with
$$\Delta^\sigma_i = [\chi^\sigma_i]_\pi + \xi^\sigma_i \cdot \pi - \mathsf{CRT}_{p_1,\ldots,p_\ell}(2r''_{i,1} + s_{1,i}, \ldots, 2r''_{i,\ell} + s_{\ell,i}),$$
where $r''_{i,j} \leftarrow \mathbb{Z} \cap [0, 2^\rho)$ and $\xi^\sigma_i \leftarrow \mathbb{Z} \cap [0, 2^{\lambda+\ell\cdot\eta}/\pi)$.

Output the secret key $\mathsf{sk} = (\mathbf{s}_0, \ldots, \mathbf{s}_\ell)$ and the public key
$$\mathsf{pk} = \left\langle x_0, \mathsf{se}_1, (\Delta_i)_{1\leqslant i\leqslant\tau}, (\Delta'_i)_{1\leqslant i\leqslant\ell}, \mathsf{se}_2, u_1, \ldots, u_\ell, \mathsf{se}_3, (\Delta^\sigma_i)_{1\leqslant i\leqslant\Theta} \right\rangle.$$

$\mathsf{BDGHV.Encrypt}(\mathsf{pk}, \mathbf{m} \in \{0,1\}^\ell)$. Use $f_1(\mathsf{se}_1)$ to recover the integers χ_i's and χ'_i's. Let $x_i = \chi_i - \Delta_i$ for $1 \leqslant i \leqslant \tau$ and $x'_i = \chi'_i - \Delta'_i$ for $1 \leqslant i \leqslant \ell$. Choose a random integer vector $\mathbf{b} = (b_i)_{1\leqslant i\leqslant\tau} \in [0, 2^\beta)^\tau$ and output the ciphertext:
$$c = \left[\sum_{i=0}^{\ell} m_i \cdot x'_i + 2 \cdot \sum_{i=1}^{\tau} b_i \cdot x_i \right]_{x_0}.$$

$\mathsf{BDGHV.Add}(\mathsf{pk}, c_1, c_2)$. Output $c_1 + c_2 \bmod x_0$

$\mathsf{BDGHV.Mult}(\mathsf{pk}, c_1, c_2)$. Output $c_1 \cdot c_2 \bmod x_0$.

$\mathsf{BDGHV.Expand}(\mathsf{pk}, c)$. The ciphertext expand procedure takes a ciphertext c and compute the associated expanded ciphertext. To do so, use $f_2(s_2)$ to recover $u_{\ell+1}, \ldots, u_\Theta$, then let $y_i = u_i/2^\kappa$ and compute z_i given by
$$z_i = \lfloor c \cdot y_i \rceil \bmod 2$$
with n bits of precision after the binary point. Define the vector $\mathbf{z} = (z_i)_{i=1,\ldots,\Theta}$ and output the expanded ciphertext $(c, \mathbf{z})$.

BDGHV.Decrypt($\mathsf{sk}, c, \mathbf{z}$). Output $\mathbf{m} = (m_1, \ldots, m_\ell)^t$ with

$$m_j \leftarrow \left[\left\lfloor \sum_{i=1}^{\Theta} s_{j,i} \cdot z_i \right\rceil \right]_2 \oplus (c \bmod 2).$$

BDGHV.Recrypt($\mathsf{pk}, c, \mathbf{z}$). Apply the decryption circuit to the expanded ciphertext $\mathbf{z}$, and the encrypted secret key bits σ_i. Output the result as a refreshed ciphertext c_{new}.

This concludes the complete description of our fully homomorphic encryption scheme with batch feature. Let us recall the parameters mentioned in Section 7.3.3:

$$\rho = \tilde{\mathcal{O}}(\lambda), \quad \eta = \tilde{\mathcal{O}}(\lambda^2), \quad \gamma = \tilde{\mathcal{O}}(\lambda^5), \quad \alpha = \tilde{\mathcal{O}}(\lambda^2), \quad \tau = \tilde{\mathcal{O}}(\lambda^3),$$

and

$$\Theta = \tilde{\mathcal{O}}(\lambda^3), \quad \theta = \lambda, \quad \ell = \tilde{\mathcal{O}}(\lambda^2).$$

For each individual x_i, x'_i and σ_i the size of the correction is $(\ell \cdot \eta + \lambda)$. The squashing procedure adds incompressible additional terms $u_1, \ldots, u_\ell$ of size $\gamma = \tilde{\mathcal{O}}(\lambda^5)$. In total we get a public key size of:

$$(\tau + \ell + \Theta) \cdot (\ell \cdot \eta + \lambda) + \ell \cdot \gamma\,.$$

Using $\tau \simeq \Theta \gg \ell$ and $\Theta \cdot \eta \simeq \gamma$, this gives a public key size:

$$2\Theta \cdot \ell \cdot \eta + \ell \cdot \gamma \approx 3\ell \cdot \gamma = \tilde{\mathcal{O}}(\lambda^7).$$

Remark 7.29. Note that, to add the possibility of perform additional permutations on the underlying plaintext, the size of the public key grows by a factor of $\Theta \cdot (\ell \cdot \eta + \lambda) \approx \ell \cdot \gamma$ for each additional permutation ζ added to the secret key. Therefore, when the σ_i^ζ's for the cycle $\zeta = (1, 2, \ldots, \ell)$ is added, the public key size becomes $4\ell \cdot \gamma$, and any set of operation is possible, from a ciphertext, on the underlying plaintext. In the homomorphic AES byte-wise bitslicing implementation of Chapter 10, the public key is roughly of size $7\ell \cdot \gamma$.

7.5.2 Semantic Security

Let us prove the semantic security of our batch DGHV scheme with compressed public key. Note that the random oracle model is only necessary when using compressed public keys as in [CNT12]; the semantic security of our batch FHE scheme from Section 7.3 does not require random oracles.

Theorem 7.30. *The previous encryption scheme is semantically secure under the decisional EF-AGCD assumption, in the random oracle model.*

sketch. The proof is almost the same as in Section 7.3.2.

More precisely, we follow the same strategy of [CNT12]: given a random oracle $H\colon \{0,1\}^* \to \mathbb{Z} \cap [0, x_0)$, we assume the pseudo-random number generation of the χ_i's and χ'_i's is defined as $\chi_i = H(\mathsf{se} \| i)$ and $\chi'_i = H(\mathsf{se} \| i + \tau)$, and we show that the integers x_i's and x'_i's generated during the BDGHV.Keygen have a distribution statistically close to their distribution in the BDGHV.Keygen of Section 7.3.2.

We can add a game $\mathbf{Game}_{-1}$ to the security proof, in which we generate a random seed se and programs the random oracle in the following way for all $1 \leqslant i < \tau + \ell$:

$$H(\mathsf{se} \| i) = \begin{cases} x_i + \Delta_i & \text{if } 1 \leqslant i \leqslant \tau \\ x'_{i-\tau} + \Delta'_{i-\tau} & \text{if } \tau + 1 \leqslant i \leqslant \tau + \ell \end{cases}$$

where $\Delta_i, \Delta'_{i-\tau} \leftarrow \mathbb{Z} \cap [0, 2^{\lambda+\eta})$. For other inputs, the oracle H is simulated in the usual way, *i.e.* by generating a random input in $[0, x_0)$ for every fresh input. Finally, we output a public key $\mathsf{pk} = \left\langle x_0, \mathsf{se}_1, (\Delta_i)_{1 \leqslant i \leqslant \tau}, (\Delta'_i)_{0 \leqslant i \leqslant \ell} \right\rangle$.

The following Lemma shows that the distribution of the public key is statistically close to that in $\mathbf{Game}_0$. This is Lemma 1 of [CNT12], where p is replaced by π and η by $\ell \cdot \eta$, and where r is draw from $(-\pi/2, \pi/2)$; since the size of r did not imply anything on the statistical distance in the proof, it is not necessary to give the proof here.

Lemma 7.31. *The following two distributions have statistical distance $\mathcal{O}(2^{-\lambda})$:*

$$\begin{aligned}
\mathcal{D} &= \big\{(\chi, \delta, x); \quad \chi \leftarrow \mathbb{Z} \cap [0, 2^{\gamma}),\ \delta = [\chi]_{\pi} + \xi \cdot \pi - r,\ x = \chi - \delta, \\
&\qquad\qquad \xi \leftarrow \mathbb{Z} \cap [0, 2^{\lambda+\ell\cdot\eta}/\pi),\ r \leftarrow \mathbb{Z} \cap (-\pi/2, \pi/2)\big\} \qquad \textit{and} \\
\mathcal{D}' &= \big\{(\chi, \delta, x); \quad x = q \cdot p + r,\ \delta \leftarrow \mathbb{Z} \cap [0, 2^{\lambda+\ell\cdot\eta}),\ \chi = x + \delta, \\
&\qquad\qquad q \leftarrow \mathbb{Z} \cap [0, 2^{\gamma}/\pi),\ r \leftarrow \mathbb{Z} \cap (-\pi/2, \pi/2)\big\}.
\end{aligned}$$

The rest of the proof follows directly. □

7.6 Implementation and Benchmarks

In this section, we explain how to select concrete parameters to instantiate the complete FHE scheme described in Section 7.5, and provide some benchmarks of our `C++` implementation on a midrange computer.

7.6.1 Practical Parameters

We use the `SAGE` [S+14] functions provided on Figures 7.1 (page 86), 7.2 (page 91) and 7.3 (page 92) to derive parameters for our FHE scheme that ensures λ bits of security.

First, we generate (ρ, η, γ) so that Chen and Nguyen's attack [CN12] and the orthogonal lattice attack takes at least 2^{λ} cycles. We provide on Figure 7.5 some `SAGE` function to achieve this. The idea is to start with an upper bound on ρ and to decrease it bit by bit until Chen and Nguyen's attack takes less than 2^{λ} cycles. As in [CMNT11, CNT12], we take $n = 4$ and $\theta = 15$ for all security levels. Therefore the degree of the decryption polynomial is $\theta = 15$ and we select $\eta = (2\theta + 8)\rho$ to allow some margin.

```
theta=15
def generate_rho_gamma_2(sec_parameter,C=158,hermite_factor=1.005,
conservative=False):
    rho = 2*sec_parameter

    while True:
        rho = rho-1
        eta = (2*theta+8)*rho
        gamma = gamma_from_orthogonal_attack_2(sec_parameter,eta,
        hermite_factor,conservative)

        if cost_CN12_attack(rho,gamma)<sec_parameter:
            rho = rho+1
            eta = (2*theta+8)*rho
            gamma = gamma_from_orthogonal_attack_2(sec_parameter,eta,
            hermite_factor,conservative)
            return rho,eta,gamma
```

Figure 7.5 – `SAGE` function to generate ρ, η and γ for λ bits of security.

Next, we select $\Theta = m_{\min}$ – defined in Equation (7.10) – as in [CMNT11, Section 6.3]. We select β as the largest value possible (so that correctness is ensured) and this gives the smallest value possible for τ from the condition $\beta \cdot \tau > \gamma + 2\lambda$. Then we set the maximum value of ℓ as Θ/θ (because for the bootstrapping to work, we want to have at most one non-zero value per window

in the secret key; to maximize ℓ, we increase Θ slightly to have $\Theta \bmod \theta = 0$), and we increase γ according to Corollary 7.12. The resulting SAGE function is given on Figure 7.6.

```
def parameters_2(sec_parameter,C=158,hermite_factor=1.005,conservative=False):

    rho,eta,gamma=generate_rho_gamma_2(sec_parameter,C,hermite_factor,
    conservative)

    Theta = m_min(sec_parameter, eta, gamma, hermite_factor)
    Theta = (Theta//theta+1)*theta if Theta%theta != 0 else Theta
    beta=eta-2*rho
    tau=ceil((gamma+2*sec_parameter)/beta)
    ell=Theta//theta           # maximum value for ell
    gamma=gamma+(ell-1)*eta  # we increase the value of gamma for batching

    print "lambda=",sec_parameter,"rho=",rho,"eta=",eta,"gamma=",gamma
    print "Theta=",Theta,"ell=",ell,"beta=",beta,"tau=",tau

    R2=RealField(12)
    pksize = (tau+ell+Theta)*(ell*eta+sec_parameter)+ell*gamma
    print "pksize=",R2(pksize/8/1024./1024.),"MB"
```

Figure 7.6 – SAGE function to generate BDGHV parameters for λ bits of security.

Finally, we provide in Table 7.3 concrete parameters for our multi-slot DGHV scheme.

Table 7.3 – Concrete Parameters for BDGHV.

Instance	λ	ℓ	ρ	η	$\gamma \times 10^{-6}$	τ	Θ	β	pk size
Toy	42	11	27	1026	0.170	165	165	972	0.684 MB
Small	52	39	42	1596	0.973	604	585	1512	13.6 MB
Medium	62	149	57	2166	5.11	2337	2235	2052	272 MB
Large	72	544	72	2736	23.3	8420	8160	2592	4550 MB

7.6.2 Implementation in C++ and Benchmarking

We implemented the scheme described in Section 7.5 in C++, using the GMP library. We obtain essentially the same running times as in [CNT12]. The main difference is that the Recrypt operation is now performed in parallel over $\ell = 544$ bits (for the "Large" setting) instead of a single bit. We provide our benchmarks in Table 7.4.

Table 7.4 – Benchmarking for our Batch DGHV with a compressed public key on a desktop computer (Intel Core i7 at 3.4Ghz, 32GB RAM).

Instance	λ	Keygen	Encrypt	Decrypt	Mult	Expand	Recrypt
Toy	42	0.05s	0.01s	0s	0.002s	0.005s	0.10s
Small	52	1.86s	0.20s	0.02s	0.016s	0.10s	0.89s
Medium	62	85s	4.67s	0.54s	0.16s	1.91s	13.26s
Large	72	3670s	63s	11s	0.68s	32.6s	189s

7.7 Conclusion

In this chapter, we first recalled the computational Approximate-GCD problem introduced by Howgrave-Graham [HG01], *building block* of the FHE scheme over the integers proposed by van Dijk,

Gentry, Halevi and Vaikuntanathan at Eurocrypt 2010 [vDGHV10], later improved by Coron and others [CMNT11, CNT12]. Then we focused on the error-free settings (when an exact multiple of the secret is known) and introduced a new decisional variant of the Approximate-GCD problem, that we proved to be equivalent to the computational Approximate-GCD problem. Then, we reviewed known attacks on the Approximate-GCD problem and provided some SAGE [S$^+$14] algorithms to select parameters ensuring λ bits of security against these attacks.

Thanks to the decisional variant of the Approximate-GCD, we simplified the original FHE scheme over the integers [vDGHV10], and described a more general variant in which the plaintext space is $\mathbb{Z}_g$. This one-slot DGHV scheme is proved to be secure against the decisional Error-Free Approximate-GCD problem.

Next, we added to this scheme a batching capability, *i.e.* we proposed a multi-slot DGHV scheme, whose security relies on the same hardness problem as for the one-slot DGHV scheme. We also showed how to perform arbitrary permutations on the underlying plaintext vector given the ciphertext and the public key, and we recall how to make these schemes fully homomorphic by squashing the decryption circuit and homomorphically evaluating it. Additionally, when considered as a somewhat homomorphic encryption scheme without bootstrapping, we have that our plaintext space is $\mathcal{M} = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_\ell}$ and is therefore adapted, by the Chinese Remainder Theorem, to perform homomorphic computations *over the integers* (*i.e.* the plaintexts are integers to be homomorphically multiplied or added over $\mathbb{Z}$, as long as the result is smaller than $g_1 \times \cdots \times g_n$). This feature is not known to be possible for other FHE constructions.

For the sake of completeness, we provided the complete description of the resulting multi-slot DGHV scheme with the public key compression technique introduced in [CNT12] and proved that this scheme remains secure in the random oracle model.

Finally, we implemented the multi-slot DGHV scheme in C++ using the big integer library GMP, and obtained running times similar to the one-slot variant [CNT12] but where bit-vectors are encrypted instead of single bits. This implementation will be later used in Chapter 7 to homomorphically evaluate the AES circuit.

The Recrypt operation appears to be the (performance) bottleneck in homomorphic evaluations. To deal with this issue, we will focus on two orthogonal approaches in the next chapters. In Chapter 8, we study how to exponentially reduce the noise growth in the multiplicative depth of the circuit. And in Chapter 9, we study how to compute the *minimal* number of Recrypt required to evaluate any given circuit.

Chapter 8

Scale-Invariant Fully Homomorphic Encryption over the Integers

8.1 Introduction

At Crypto 2012, Brakerski constructed a scale-invariant fully homomorphic encryption scheme based on the LWE problem, in which the same modulus is used throughout the evaluation process, instead of a ladder of moduli when doing "modulus switching". In this chapter, we describe variants of the DGHV schemes of Section 7.3 with the same scale-invariant property. The resulting schemes have a single secret modulus whose size is linear in the multiplicative depth of the circuit to be homomorphically evaluated, instead of exponential in Chapter 7; we therefore construct a leveled fully homomorphic encryption scheme. This scheme can be transformed into a pure fully homomorphic encryption scheme using bootstrapping, and its security is still based on the Approximate-GCD problem.

This chapter is essentially constituted of the article Scale-Invariant Fully Homomorphic Encryption over the Integers [CLT14a], cosigned with J.-S. Coron and M. Tibouchi, and published at PKC 2014 [Kra14]. The full version of the article is available at [CLT14b].

Modulus Switching and Scale Invariance. In order to avoid bootstrapping, a new noise management technique, called *modulus switching*, was introduced by Brakerski, Gentry and Vaikuntanathan [BGV12]. The authors obtained a somewhat homomorphic FHE scheme in which the noise grows *linearly* with the multiplicative depth instead of exponentially as the initial somewhat homomorphic encryption schemes (as in Chapter 7). Therefore any circuit with polynomial depth can be evaluated. The technique consists in scaling down the noise by converting a ciphertext modulo q into a ciphertext modulo a smaller q'; the noise being reduced by roughly a factor q/q'. By carefully calibrating the ladder of moduli, the noise growth can then be made linear with the number of homomorphic multiplications. Unfortunately, because of the dimension reduction technique, for a circuit with L layers of multiplication, the technique requires to store the equivalent of L public-keys, yielding a huge storage requirement (*cf.* especially [GHS12c]).[1] The technique was also adapted to the DGHV fully homomorphic encryption scheme over the integers of [vDGHV10] (*i.e.* the one-slot DGHV scheme in Chapter 7) in [CNT12].

At Crypto 2012, Brakerski introduced a new tensor product technique for LWE-based leveled FHE [Bra12] so that the *same* modulus is used throughout the evaluation process instead of a layer of moduli; the noise growth is still linear in the number of homomorphic multiplications. This was achieved by considering ciphertexts such that $\langle \mathbf{c}, \mathbf{s} \rangle = \lfloor q/2 \rfloor \cdot m + e \bmod q$ (as in Regev's encryption scheme [Reg09]), instead of $\langle \mathbf{c}, \mathbf{s} \rangle = m + 2e \bmod q$.

Our Contributions. In this chapter, we describe a variant of the DGHV schemes over the integers proposed in Chapter 7 with the same scale-invariant property as in [Bra12]; *i.e.* our schemes do not

[1] Note that under an additional circular security assumption (that is informally the assumption that the scheme remains secure even when the adversary is given encryptions of the individual bits of the private key), the secret keys may all be the same, and therefore one can obtain a public key of size independent of L [BGV12, Section 5.5].

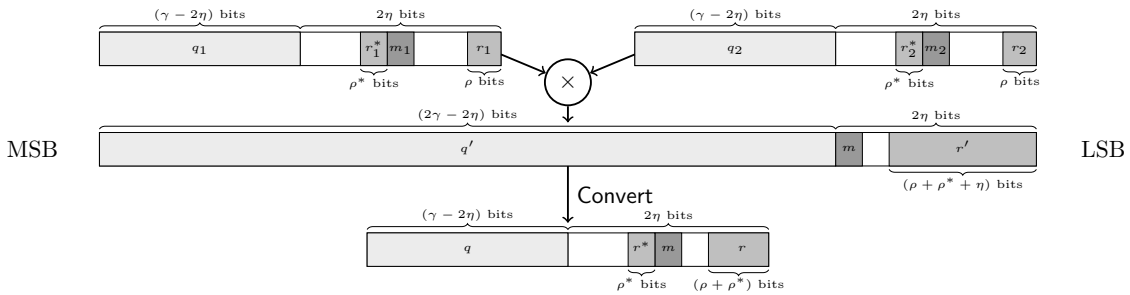


Figure 8.1 – Conversion of a ciphertext after a homomorphic multiplication.

use modulus switching and the noise grows linearly with the multiplicative depth of the circuit. We obtain DGHV variants with a single secret modulus p whose size is linear in the multiplicative depth (instead of exponential). Our technique is as follows.

In the original DGHV scheme, a ciphertext c of the bit message $m \in \{0, 1\}$ has the form

$$c = m + 2r + q \cdot p \,,$$

where p is the secret key, q is a large random integer, and r is a small random integer (noise). The bit message is recovered by computing $m = (c \bmod p) \bmod 2$. Adding and multiplying ciphertexts over $\mathbb{Z}$ respectively adds and multiplies the plaintexts modulo 2 while keeping them hidden. Unfortunately, the noise grows exponentially with the number of homomorphic multiplications: if two ciphertexts c_1, c_2 have ρ-bit noise, the noise of $c_3 = c_1 \cdot c_2$ has $\approx 2\rho$ bits. Therefore to evaluate a circuit with L sequential layers of multiplications without bootstrapping, the bit-size η of the modulus p must satisfy $\eta > 2^L \rho$.

In our new scheme, similar to [Bra12], instead of encrypting the bit $m \in \{0, 1\}$ in the LSB of $[c \bmod p]$, we encrypt it in the MSB of $[c \bmod p]$; additionally we work modulo p^2 instead of modulo p.[2] More precisely, the message m is now encrypted as

$$c = r + (m + 2r^*) \cdot \frac{p-1}{2} + q \cdot p^2 \,, \tag{8.1}$$

where the ciphertext now contains two noises r and r^*. We decrypt c by computing $m = (2c \bmod p) \bmod 2$. Clearly adding two ciphertexts over $\mathbb{Z}$ still adds the underlying bit messages m modulo 2. However, multiplication of two ciphertexts moves the bit message m from the MSB of $[c \bmod p]$ to the MSB of $[c \bmod p^2]$. Namely, a ciphertext c obtained as the multiplication of ciphertexts c_1 and c_2 for the respective bit messages m_1 and m_2 will have the form

$$c = 2 \cdot c_1 \cdot c_2 = r + (m_1 \cdot m_2) \cdot \frac{p^2-1}{2} + q \cdot p^2 \,, \tag{8.2}$$

where $r > p$ but still $r \ll p^2$. We then describe a procedure Convert that allows to publicly convert the result of a multiplication (*i.e.* a ciphertext as in Equation (8.2)) into a ciphertext reusable in subsequent homomorphic operations (*i.e.* a ciphertext as in Equation (8.1)), either keeping the same secret p (which requires, as usual, a circular security assumption – *cf.* Section 7.4.2) or using a different fresh p at each level (which requires a larger secret key [CNT12]). The bit length of the noise in the new ciphertext grows only by a constant additive factor with respect to the noise in c_1 and c_2 (see Figure 8.1 for an illustration). Therefore, our scheme is a variant of the DGHV scheme that is a leveled fully homomorphic encryption scheme. It can be turned into a pure FHE scheme using bootstrapping (*cf.* [vDGHV10, CMNT11, CNT12] and Section 7.4). We also show that our scheme is semantically secure, under the Approximate-GCD assumption.

We also adapt our scale-invariant technique to the batch setting, *i.e.* we describe how to adapt our technique to the multi-slot DGHV scheme of Section 7.3.2.

[2] Notice that we cannot work with $c = (p-1)/2 \cdot m + r + q \cdot p$ directly.

Remark 8.1. Note that in this chapter, we work with an error-free element, *i.e.* an exact multiple of p^2. In the article [CLT14a, CLT14b] corresponding to this chapter, we describe the schemes without this error-free element, as in [vDGHV10], but provide only security proofs in the error-free case.

8.2 Scale-Invariant One-Slot DGHV Scheme

In this section we describe our variant of the DGHV scheme of Section 7.3.1 with the scale-invariant property. We first explain the two main ideas of our scheme, namely (1) moving the plaintext bit from the LSB to the MSB of $[c \bmod p]$ and working modulo p^2, and (2) converting the result of a ciphertext multiplication back to a ciphertext usable in subsequent homomorphic operations. We then provide the full description of our scheme.

Throughout the section, denote by $x_0 = q_0 \cdot p^2$ the public modulus.

8.2.1 Ciphertexts and Homomorphic Operations

As explained in the introduction, instead of encrypting the plaintext $m \in \{0, 1\}$ in the LSB of $[c \bmod p]$, m is now encrypted in the MSB of $[c \bmod p]$ as

$$c = r + (m + 2r^*) \cdot \frac{p-1}{2} + q \cdot p^2 \,, \tag{8.1}$$

where the ciphertext has now two noises r and r^* of respective bit-length ρ and ρ^*. We call such ciphertext a *Type-I ciphertext* and we say that c has noise length (ρ, ρ^*). To decrypt c, one computes $(2c \bmod p) \bmod 2 = m$.

Homomorphic additions are performed as additions modulo x_0: namely given two Type-I ciphertexts c_1 and c_2 of noise (ρ, ρ^*):

$$\begin{aligned} c_1 &= r_1 + (m_1 + 2r_1^*) \cdot (p-1)/2 + q_1 \cdot p^2 \\ c_2 &= r_2 + (m_2 + 2r_2^*) \cdot (p-1)/2 + q_2 \cdot p^2 \end{aligned}$$

we get

$$c_1 + c_2 \bmod x_0 = r_3 + (m_1 + m_2 + 2r_3^*) \cdot \frac{p-1}{2} + q_3 \cdot p^2 \,,$$

for some integers r_3, r_3^* and q_3, with $\log_2 |r_3| \leqslant \rho + 1$ and $\log_2 |r_3^*| \leqslant \rho^* + 1$.

Next, to homomorphically multiply the ciphertexts c_1 and c_2, one computes $c_3 = 2 \cdot c_1 \cdot c_2 \bmod x_0$. This gives

$$\begin{aligned} c_3 = 2 \cdot c_1 \cdot c_2 \bmod x_0 &= 2r_1 r_2 + \big(r_1(m_2 + 2r_2^*) + r_2(m_1 + 2r_1^*)\big) \cdot (p-1) + \\ &\quad (m_1 + 2r_1^*) \cdot (m_2 + 2r_2^*) \cdot \frac{(p-1)^2}{2} + q_3' \cdot p^2 \\ &= r_3' + (m_1 + 2r_1^*) \cdot (m_2 + 2r_2^*) \cdot \frac{(p-1)^2}{2} + q_3' \cdot p^2 \end{aligned}$$

for some integers q_3' and r_3', with $\log_2 |r_3'| \leqslant \eta + \rho + \rho^* + 3$, where η is the bit-size of p. We use $\eta \gg \rho, \rho^*$. Then, there exist integers r_3 and q_3 such that

$$c_3 = r_3 + m_3 \cdot \frac{p^2 - 1}{2} + q_3 \cdot p^2 \,, \tag{8.2}$$

where $m_3 = m_1 \cdot m_2$. We call an integer c verifying Equation (8.2) a *Type-II ciphertext*. The bit-length of noise r_3 satisfies $\log_2 |r_3| \leqslant \eta + \rho + \rho^* + 4$, assuming $\rho^* < \rho$. We refer to Figure 8.1 for a graphical representation of the homomorphic multiplication.

8.2.2 Conversion from Type-II Ciphertext to Type-I Ciphertext

We show that we can efficiently convert a Type-II ciphertext back to a Type-I ciphertext, using only the public-key. Our procedure Convert uses essentially the same technique as the modulus

switching technique for DGHV in [CNT12]. Namely modulus switching in [CNT12] enables to convert a classical DGHV ciphertext modulo a prime p into a new ciphertext modulo a prime p', with noise scaled by a factor p'/p. Similarly, our Convert procedure converts a Type-II ciphertext modulo p^2 back to a ciphertext where the noise is modulo p (therefore the noise is scaled by a factor $p/p^2 = 1/p$), but still somehow encrypted modulo p^2.

More precisely, we start from a Type-II ciphertext:

$$c = r + \frac{p^2 - 1}{2} \cdot m + q \cdot p^2 \tag{8.3}$$

where $|r| \leqslant 2^{\rho'}$. Let κ be such that $|c| < 2^{\kappa}$. Let $\mathbf{z}$ be a vector of Θ rational numbers in $[0, 2^{\eta})$ with κ bits of precision after the binary point, and let $\mathbf{s}$ be a vector of Θ bits such that

$$\frac{2^{\eta}}{p^2} = \langle \mathbf{s}, \mathbf{z} \rangle + \varepsilon \mod 2^{\eta}, \tag{8.4}$$

where $|\varepsilon| \leqslant 2^{-\kappa}$. Here Θ is a parameter to be chosen later for security. We use the same BitDecomp and PowersofTwo procedures as in [BGV12].

- $\mathsf{BitDecomp}_{\eta}(\mathbf{v})$: For $\mathbf{v} \in \mathbb{Z}^n$, let $\mathbf{v}_i \in \{0,1\}^n$ be such that $\mathbf{v} \bmod 2^{\eta} = \sum_{i=0}^{\eta-1} \mathbf{v}_i \cdot 2^i$. Output the vector
$$(\mathbf{v}_0, \ldots, \mathbf{v}_{\eta-1})^t \in \{0,1\}^{n \cdot \eta}.$$
- $\mathsf{PowersofTwo}_{\eta}(\mathbf{w})$: For $\mathbf{w} \in \mathbb{Z}^n$, output the vector
$$(\mathbf{w}, 2 \cdot \mathbf{w}, \ldots, 2^{\eta-1} \cdot \mathbf{w})^t \in \mathbb{Z}^{n \cdot \eta}.$$

Given the vector $\mathbf{s}$ from (8.4), we let $\mathbf{s}' = \mathsf{PowersofTwo}_{\eta}(\mathbf{s})$, and let

$$\boldsymbol{\sigma} = \mathbf{q} \cdot p^2 + \mathbf{r} + \left\lfloor \mathbf{s}' \cdot \frac{p}{2^{\eta+1}} \right\rceil \tag{8.5}$$

be an "encryption" of the vector $\mathbf{s}'$, where $\mathbf{q} \leftarrow (\mathbb{Z} \cap [0, 2^{\gamma}/p^2))^{\eta \cdot \Theta}$ and $\mathbf{r} \leftarrow (\mathbb{Z} \cap (-2^{\rho}, 2^{\rho}))^{\eta \cdot \Theta}$. We can now define the Convert algorithm:

$\mathsf{Convert}(\mathbf{z}, \sigma, c)$. Compute $\mathbf{c} = (\lfloor c \cdot z_i \rceil \bmod 2^{\eta})_{1 \leqslant i \leqslant \Theta}$ and its decomposition $\mathbf{c}' = \mathsf{BitDecomp}_{\eta}(\mathbf{c})$. Then output

$$c' \leftarrow 2\langle \boldsymbol{\sigma}, \mathbf{c}' \rangle.$$

The following Lemma shows that our procedure Convert enables one to transform a Type-II ciphertext back to a Type-I ciphertext. We provide the proof in the next section.

Lemma 8.2. *Let ρ' be such that $\rho' \geqslant \eta + \rho + \log_2(\eta\Theta)$. The procedure* Convert *above converts a Type-II ciphertext with noise size ρ' into a Type-I ciphertext with noise $(\rho' - \eta + 5, \log_2 \Theta)$.*

Assume that initially the two ciphertexts c_1, c_2 are Type-I ciphertexts with noise $(\rho_1, \log_2 \Theta)$. After computing $c_3 = 2 \cdot c_1 \cdot c_2 \bmod x_0$ which has noise size at most $\rho' = \eta + \rho_1 + \log_2 \Theta + 4$ (see previous section) one can convert c_3 back into a Type-I ciphertext with noise (ρ_3, ρ_3^*) with $\rho_3 = \rho_1 + \log_2 \Theta + 9$ and $\rho_3^* = \log_2 \Theta$, if the condition of Lemma 8.2 is verified. Therefore the noise length in bits has only grown by an additive factor $\log_2 \Theta + 9$. Therefore the ciphertext noise grows only linearly with the number of homomorphic multiplications.

Remark 8.3. To make public conversion of ciphertexts possible, one has to publish $\boldsymbol{\sigma}$, which is an "encryption" of the secret key dependent vector $\mathbf{s}'$. As a result, one has to assume circular security of the underlying encryption scheme, as usual in constructions of FHE – *cf.* also Section 7.4.2.

Alternatively, as in other modulus switching-based schemes, it is also possible to define $\boldsymbol{\sigma}$ as an encryption of $\mathbf{s}'$ under a fresh secret key p', in which case $\mathsf{Convert}(\mathbf{z}, \sigma, c)$ yields a Type-I ciphertext under p'. Defining a different secret prime for each level of multiplication and publishing the corresponding conversion vectors $\boldsymbol{\sigma}$ makes it possible to avoid circular security assumptions, but of course it increases public key size, and it is also less convenient insofar as homomorphic operations are only supported between ciphertexts at the same level.

8.2.3 Proof of Lemma 8.2

We start from a Type-II ciphertext as given by Equation (8.3) with $|r| \leqslant 2^{\rho'}$. From

$$\boldsymbol{\sigma} = p^2 \cdot \mathbf{q} + \mathbf{r} + \left\lfloor \mathbf{s}' \cdot \frac{p}{2^{\eta+1}} \right\rceil$$

we have:

$$c' = 2\langle \boldsymbol{\sigma}, \mathbf{c}' \rangle = 2p^2 \cdot \langle \mathbf{q}, \mathbf{c}' \rangle + 2\langle \mathbf{r}, \mathbf{c}' \rangle + 2 \left\langle \left\lfloor \mathbf{s}' \cdot \frac{p}{2^{\eta+1}} \right\rceil, \mathbf{c}' \right\rangle . \tag{8.6}$$

Since the components of $\mathbf{c}'$ are bits, we have using $2\lfloor x/2 \rceil = x + \nu$ with $|\nu| \leqslant 1$:

$$2 \left\langle \left\lfloor \frac{p}{2^{\eta+1}} \cdot \mathbf{s}' \right\rceil, \mathbf{c}' \right\rangle = \left\langle \frac{p}{2^{\eta}} \cdot \mathbf{s}', \mathbf{c}' \right\rangle + \nu_2 = \frac{p}{2^{\eta}} \cdot \langle \mathbf{s}', \mathbf{c}' \rangle + \nu_2 , \tag{8.7}$$

where $|\nu_2| \leqslant \Theta \cdot \eta$. From the definition of BitDecomp and PowersofTwo, we have $\langle \mathbf{s}', \mathbf{c}' \rangle = \langle \mathbf{s}, \mathbf{c} \rangle \bmod 2^{\eta} = \langle \mathbf{s}, \mathbf{c} \rangle + q_2 \cdot 2^{\eta}$ with $q_2 \in \mathbb{Z}$. Moreover

$$\langle \mathbf{s}, \mathbf{c} \rangle = \sum_{i=1}^{\Theta} s_i \lfloor c \cdot z_i \rceil + \Delta \cdot 2^{\eta} = \sum_{i=1}^{\Theta} s_i \cdot c \cdot z_i + \delta_1 + \Delta \cdot 2^{\eta} = c \cdot \langle \mathbf{s}, \mathbf{z} \rangle + \delta_1 + \Delta \cdot 2^{\eta} ,$$

for some $\Delta \in \mathbb{Z}$ and $|\delta_1| \leqslant \Theta/2$. Using $\langle \mathbf{s}, \mathbf{z} \rangle = 2^{\eta}/p^2 - \varepsilon - \mu \cdot 2^{\eta}$ for some $\mu \in \mathbb{Z}$, and $c = q \cdot p^2 + m \cdot (p^2 - 1)/2 + r$, this gives

$$\langle \mathbf{s}, \mathbf{c} \rangle = c \cdot \left(\frac{2^{\eta}}{p^2} - \varepsilon - \mu \cdot 2^{\eta} \right) + \delta_1 + \Delta 2^{\eta} = q \cdot 2^{\eta} + m \cdot 2^{\eta-1} - m \cdot \frac{2^{\eta}}{2p^2} + r \cdot \frac{2^{\eta}}{p^2} - c \cdot \varepsilon + \delta_1 + (\Delta - c \cdot \mu) \cdot 2^{\eta} .$$

Therefore we can write

$$\langle \mathbf{s}, \mathbf{c} \rangle = q_1 \cdot 2^{\eta} + m \cdot 2^{\eta-1} + r^*$$

for some $r^* \in \mathbb{Z}$, with $|r^*| \leqslant 2^{\rho'-\eta+3}$ (because Θ is small). We get from Equation (8.7):

$$2 \left\langle \left\lfloor \frac{p}{2^{\eta+1}} \cdot \mathbf{s}' \right\rceil, \mathbf{c}' \right\rangle = \frac{p}{2^{\eta}} \cdot ((q_1 + q_2) \cdot 2^{\eta} + m \cdot 2^{\eta-1} + r^*) + \nu_2 = q_3 \cdot p + m \cdot \frac{p}{2} + \frac{p}{2^{\eta}} \cdot r^* + \nu_2 ,$$

with $|q_3| \leqslant \Theta$; namely the components of $(p/2^{\eta+1}) \cdot \mathbf{s}'$ are smaller than p and $\mathbf{c}'$ is a binary vector. This gives

$$2 \left\langle \left\lfloor \frac{p}{2^{\eta+1}} \cdot \mathbf{s}' \right\rceil, \mathbf{c}' \right\rangle = (2q_3 + m) \cdot \frac{p-1}{2} + r_2^* ,$$

with again $|r_2^*| \leqslant 2^{\rho'-\eta+4}$. Therefore we obtain from Equation (8.6):

$$\begin{aligned} c' &= 2p^2 \cdot \langle \mathbf{q}, \mathbf{c}' \rangle + 2\langle \mathbf{r}, \mathbf{c}' \rangle + (2q_3 + m) \cdot \frac{p-1}{2} + r_2^* \\ c' &= 2q'' \cdot p^2 + (2q_3 + m) \cdot \frac{p-1}{2} + r' . \end{aligned}$$

where $|r'| \leqslant |r_2^*| + \eta\Theta 2^{\rho+1} \leqslant 2^{\rho'-\eta+4} + \eta\Theta 2^{\rho+1}$, which proves the lemma (using the fact that $\rho' \geqslant \eta + \rho + \log_2(\eta\Theta)$).

8.2.4 Description of the Public-Key Leveled Homomorphic Scheme

We are now ready to describe our scale-invariant version of the DGHV encryption scheme.

SIDGHV.Keygen(1^{λ}). Generate a 2^{ν}-rough η-bit integer p and randomly generate a 2^{ν}-rough integer $q_0 \in [0, 2^{\gamma}/p^2)$. Denote $x_0 = p^2 \cdot q_0$. Next sample τ integers $\{x_i\}_{i=1,\ldots,\tau}$ from the AGCD distribution $D_{\rho}(p^2, q_0)$.

Sample also an integer y from the shifted AGCD distribution $D_{\rho}(p^2, q_0) + \frac{p-1}{2}$.

Let $\mathbf{z}$ be a vector of Θ numbers with $\kappa = \gamma$ bits of precision after the binary point, and let $\mathbf{s}$ be a vector of Θ bits such that

$$\frac{2^{\eta}}{p^2} = \langle \mathbf{s}, \mathbf{z} \rangle + \varepsilon \bmod 2^{\eta} ,$$

with $|\varepsilon| \leqslant 2^{-\kappa}$. Now, define

$$\sigma = \mathbf{q} \cdot p^2 + \mathbf{r} + \left\lfloor \mathsf{PowersofTwo}_\eta(\mathbf{s}) \cdot \frac{p}{2^{\eta+1}} \right\rceil ,$$

where the components of $\mathbf{q}$ (resp. $\mathbf{r}$) are randomly chosen from $[0, q_0) \cap \mathbb{Z}$ (resp. $[0, 2^\rho) \cap \mathbb{Z}$). Let $\mathsf{sk} = p$ and $\mathsf{pk} = \{x_0, x_1, \ldots, x_\tau, y, \sigma, \mathbf{z}\}$.

$\mathsf{SIDGHV.Encrypt}(\mathsf{pk}, m \in \{0,1\})$. Generate a random β-bit integer vector $\mathbf{b} = (b_1, \ldots, b_\tau)^t$ and output

$$c \leftarrow \left(m \cdot y + \sum_{i=1}^{\tau} b_i \cdot x_i \right) \bmod x_0 .$$

$\mathsf{SIDGHV.Add}(\mathsf{pk}, c_1, c_2)$. Output $c \leftarrow (c_1 + c_2) \bmod x_0$.

$\mathsf{SIDGHV.Convert}(\mathsf{pk}, c)$. Output $c' \leftarrow 2 \cdot \left\langle \sigma, \mathsf{BitDecomp}_\eta(\mathbf{c}) \right\rangle \bmod x_0$ where $\mathbf{c} = \left(\lfloor c \cdot z_i \rceil \bmod 2^\eta \right)_{1 \leqslant i \leqslant \Theta}$.

$\mathsf{SIDGHV.Mult}(\mathsf{pk}, c_1, c_2)$. Output $c' \leftarrow \mathsf{SIDGHVConvert}(\mathsf{pk}, 2 \cdot c_1 \cdot c_2 \bmod x_0)$.

$\mathsf{SIDGHV.Decrypt}(\mathsf{sk}, c)$. Output $m \leftarrow \big((2c) \bmod p\big) \bmod 2$.

Remark 8.4. This describes a *leveled* fully homomorphic encryption scheme, because the noise growth is only linear in the number of levels. The scheme can be bootstrapped to obtain a (pure) fully homomorphic encryption scheme.

8.2.5 Constraints on the Parameters

The parameters of the scheme must basically meet the same constraints as in Section 7.3.3, Table 7.2. In particular, if λ is the security parameter:

- $\rho = \Omega(\lambda)$ to avoid brute force attack on the noise [CN12, CNT12],
- $\eta \geqslant \beta + \rho + \log_2(\tau + 1) + 1 + \mathcal{O}(L \log \lambda)$ where L is the multiplicative depth of the circuit to be evaluated,
- $\gamma = \eta^2 \cdot \Omega(\lambda)$ in order to thwart lattice-based attacks,
- $\Theta^2 = \gamma \cdot \Omega(\lambda)$ to avoid lattice attacks on the subset sum (see [CMNT11]),
- $\beta \cdot \tau \geqslant \gamma + 2\lambda$ in order to apply the Leftover Hash Lemma in the security proof.

To satisfy the above constraints one can take $\rho = 2\lambda$, $\beta = \tilde{\mathcal{O}}(L + \lambda)$, $\eta = \tilde{\mathcal{O}}(L + \lambda)$, $\gamma = \tilde{\mathcal{O}}(L^2\lambda + \lambda^3)$, $\Theta = \tilde{\mathcal{O}}(\sqrt{\lambda} \cdot (L + \lambda))$ and $\tau = \tilde{\mathcal{O}}(L^2 + \lambda^2)$.

8.2.6 Semantic Security

We show that the semantic security of our scheme can be based on the following variant of the decisional Error-Free Approximate-GCD problem introduced in Definition 7.4.

Definition 8.5. Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$.

The *decisional squared Error-Free-AGCD with additional element* problem is: For a 2^ν-rough η-bit integer p and a uniformly chosen 2^ν-rough $q_0 \in [0, 2^\gamma / p^2)$, given $x_0 = q_0 \cdot p^2$, a sample y from $(p-1)/2 + D_\rho(p^2, q_0)$ and polynomially many samples $\{x_i\}_i$ from $\mathbb{Z}_{x_0}$ to distinguish whether the samples $\{x_i\}_i$ are distributed uniformly or whether they are distributed according to the AGCD distribution $D_\rho(p^2, q_0)$.

Note that the decisional squared EF-AGCD problem reduces to the decision EF-AGCD problem. Indeed, the input of the former problem with parameters $(\gamma, \eta, \nu, \rho)$, without the additional element y, is an input of the latter problem with parameters $(\gamma, 2\eta, \nu, \rho)$.

The following theorem shows that our scheme is semantically secure under the decisional squared Error-Free-AGCD with additional element assumption; below we only consider a subset of our scheme without the procedure Convert, *i.e.* without the public parameters $\mathbf{z}$ and $\boldsymbol{\sigma}$. To prove the semantic security of the full scheme it suffices to include $\mathbf{z}$ and $\boldsymbol{\sigma}$ in the above decisional assumption.[3]

Theorem 8.6. *The above scale-invariant DGHV scheme without the parameters $\mathbf{z}$, $\boldsymbol{\sigma}$ is semantically secure under the $(\gamma, \eta, \nu, \rho)$-decisional squared Error-Free-AGCD with additional element assumption.*

The proof of this theorem is similar to the proof of semantic security, Theorem 7.17 in Chapter 7. First, we have the following lemma (similarly to Lemma 7.18):

Lemma 8.7. *For the parameters $(\gamma, \eta, \nu, \rho)$, let $\mathsf{pk} = (x_0, \{x_i\}_i, y)$ and $\mathsf{sk} = p$ be chosen as in the SIDGHV.Keygen procedure. Define $\mathsf{pk}' = (x_0, \{x'_i\}_i, y)$ for x'_i uniformly generated in $[0, x_0)$. Then pk and pk' are indistinguishable under the decisional squared Error-Free-Approximate-GCD with additional element assumption.*

Then the proof of Theorem 8.6 is exactly the proof of Theorem 7.17, using Lemma 8.7 instead of Lemma 7.18.

Remark 8.8. The equivalence between the error-free decisional Approximate-GCD and error-free computational Approximate-GCD assumption, proved in Section 7.2.2, is straightforwardly adaptable to a computational squared Error-Free-AGCD with additional element problem, which aims at recovering p from the additional element y and samples from $D_\rho(p^2, q_0)$.

Remark 8.9. We could not show an equivalence between the decisional Error-Free Approximate-GCD assumption (Definition 7.4), and the decisional squared Error-Free-Approximate-GCD with additional element assumption. However, the knowledge of y does not seem to simplify the attacks against the Approximate-GCD problem proposed in Section 7.2.4 and we are not aware of any attack that would use such an additional element.

8.3 Scale-Invariant Multi-Slot DGHV Scheme

We now describe a generalization of the previous scheme to the batch setting, *i.e.* to the multi-slot DGHV scheme presented in Section 7.3.2. The goal is to pack ℓ plaintext bits $m_1, \ldots, m_\ell$ into a single ciphertext. Homomorphic addition and multiplication will then apply in parallel and component-wise on the m_i's.

Our batch generalization is similar to the one of Chapter 7. A ciphertext encrypting a vector $\mathbf{m} = (m_1, \ldots, m_\ell)^t$ has the form:

$$c = \mathsf{CRT}_{q_0, p_1^2, \ldots, p_\ell^2}\left(q, \ldots, r_i + (2r_i^* + m_i) \cdot \frac{p_i - 1}{2}, \ldots\right) \tag{8.8}$$

for a tuple of $\ell + 1$ coprime integers $q_0, p_1, \ldots, p_\ell$. We call such ciphertext a *batch Type-I ciphertext*. Modulo each of the p_j's the ciphertext c behaves as in the SIDGHV scheme in Section 8.2. In the following, denote $x_0 = q_0 \cdot \prod_i p_i^2$ the public modulus. Accordingly, the addition of two ciphertexts modulo x_0 yields a new ciphertext that decrypts to the componentwise sum modulo 2 of the original plaintexts.

[3] Usually in FHE we first show the semantic security of a restricted scheme, and then a 'circular security' assumption is used to get the semantic security of the entire FHE; that is we assume that the encryption scheme remains secure even when the adversary is given encryptions of the individual bits of the private-key.

Here we first prove that the scheme is secure without the terms $\mathbf{z}$ and σ. If the scheme is 'circular secure' (secure even with encryptions of the invariant switching, *i.e.* $\mathbf{z}$ and σ) then it remains semantically secure. This circular security assumption can be avoided by using the classical modulus switching technique [CNT12] instead of our scale-invariance technique.

To homomorphically multiply two ciphertexts c_1 and c_2, one computes $c_3 = 2 \cdot c_1 \cdot c_2 \bmod x_0$. As previously there exists small integers $r_{3,j}$ such that

$$c_3 \equiv r_{3,j} + m_j \cdot \frac{p_j^2 - 1}{2} \pmod{p_j} \quad \text{for } j = 0, \ldots, \ell - 1, \tag{8.9}$$

where each m_j is the product of the corresponding plain text components of c_1 and c_2. We call c_3 a *batch Type-II ciphertext*. Modulo each of the p_j's, the ciphertext c_3 behaves as a Type-II ciphertext given by Equation (8.2); therefore the message bit m_j is the MSB of $[c \bmod p_j^2]$ for all j. As in Section 8.2, there exists an efficient conversion procedure $\mathsf{Convert}$ to convert any Type-II ciphertext to a new Type-I ciphertext. As shown below the procedure $\mathsf{Convert}$ is actually the same as in Section 8.2, with adapted public parameters.

Namely let $\mathbf{z}$ be a vector of Θ rational numbers in $[0, 2^\eta)$ with κ bits of precision after the binary point (where $|c| < 2^\kappa$), and let $(\mathbf{s}_j)$ be a set of ℓ vectors of Θ bits such that, for all $j = 1, \ldots, \ell$,

$$\frac{2^\eta}{p_j^2} = \langle \mathbf{s}_j, \mathbf{z} \rangle + \varepsilon_j \mod 2^\eta$$

where $|\varepsilon_j| \leqslant 2^{-\kappa}$. Let $\mathbf{s}'_j = \mathsf{PowersofTwo}_\eta(\mathbf{s}_j) \in \mathbb{Z}^{\eta\Theta}$. Define $\sigma = (\sigma_1, \ldots, \sigma_{\eta\Theta})$ so that, for all $1 \leqslant i \leqslant \eta\Theta$:

$$\sigma_i = \mathsf{CRT}_{q_0, p_1^2, \ldots, p_\ell^2} \left(q_i, r_{1,i} + \left\lfloor s'_{1,i} \cdot \frac{p_1}{2^{\eta+1}} \right\rceil, \ldots, r_{\ell,i} + \left\lfloor s'_{\ell,i} \cdot \frac{p_\ell}{2^{\eta+1}} \right\rceil \right)$$

is an encryption of $(s'_{j,i})_{1 \leqslant j \leqslant \ell}$. For $\mathsf{Convert}$ we use the same algorithm as in Section 8.2:

$\mathsf{Convert}(\mathbf{z}, \sigma, c)$. Compute $\mathbf{c} = (\lfloor c \cdot z_i \rceil \bmod 2^\eta)_{1 \leqslant i \leqslant \Theta}$ and its decomposition $\mathbf{c}' = \mathsf{BitDecomp}_\eta(\mathbf{c})$. Then output

$$c' \leftarrow 2 \langle \boldsymbol{\sigma}, \mathbf{c}' \rangle \bmod x_0 \,.$$

The proof of the following lemma follows directly from the proof of Lemma 8.2 applied modulo each of the p_j's.

Lemma 8.10. *The procedure* $\mathsf{Convert}$ *above converts a Type-II ciphertext with noise size* ρ' *into a Type-I ciphertext with noise* $(\rho' - \eta + 5, \log_2 \Theta)$*, for* $\rho' - \eta \geqslant \rho + \log_2(\eta\Theta)$*.*

8.3.1 Description of the Public-Key Batch Leveled Fully Homomorphic Scheme

$\mathsf{SIBDGHV.Keygen}(1^\lambda)$. Generate ℓ 2^ν-rough η-bit integer p_i's and randomly generate a 2^ν-rough integer $q_0 \in [0, 2^\gamma / \pi^2)$ where $\pi = \prod_i p_i$. Denote $x_0 = \pi^2 \cdot q_0$.

Next sample τ integers $\{x_i\}_{i=1,\ldots,\tau}$ from the ℓ-AGCD distribution $D_\rho^{(\ell)}(p_1^2, \ldots, p_\ell^2, q_0)$.

Then for $1 \leqslant i \leqslant \ell$, sample y'_i from the ℓ-AGCD distribution $D_\rho^{(\ell)}(p_1^2, \ldots, p_\ell^2, q_0)$ and define

$$y_i = y'_i + \frac{p_i - 1}{2} \cdot \prod_{\substack{j=1 \\ j \neq i}}^{\ell} p_j^2 \,.$$

Let $\mathbf{z}$ be a vector of Θ numbers with $\kappa = \gamma$ bits of precision after the binary point, and let $(\mathbf{s}_j)$ be a set of ℓ vectors of Θ bits such that, for all $j = 1, \ldots, \ell$,

$$\frac{2^\eta}{p_j^2} = \langle \mathbf{s}_j, \mathbf{z} \rangle + \varepsilon_j \mod 2^\eta$$

with $|\varepsilon_j| \leqslant 2^{-\kappa}$. Let $\mathbf{s}'_j = \mathsf{PowersofTwo}_\eta(\mathbf{s}_j) \in \mathbb{Z}^{\eta\Theta}$. Then, define $\sigma = (\sigma_1, \ldots, \sigma_{\eta\Theta})$ so that, for all $1 \leqslant i \leqslant \eta\Theta$:

$$\sigma_i = \mathsf{CRT}_{q_0, p_1^2, \ldots, p_\ell^2} \left(q_i, r_{1,i} + \left\lfloor s'_{1,i} \cdot \frac{p_1}{2^{\eta+1}} \right\rceil, \ldots, r_{\ell,i} + \left\lfloor s'_{\ell,i} \cdot \frac{p_\ell}{2^{\eta+1}} \right\rceil \right)$$

where q_i (resp. $r_{j,i}$) are randomly chosen from $[0, q_0)$ (resp. $[0, 2^\rho)$).

The secret-key is $\mathsf{sk} = (p_0, \ldots, p_{\ell-1})$ and the public-key is $\mathsf{pk} = (x_0, x_1, \ldots, x_\tau, y_1, \ldots, y_\ell, \sigma, \mathbf{z})$.

$\mathsf{SIBDGHV.Encrypt}(\mathsf{pk}, \mathbf{m} \in \{0,1\}^\ell)$. Generate a random β-bit integer vector $\mathbf{b} = (b_1, \ldots, b_\tau)^t$ and output

$$c \leftarrow \Big(\sum_{i=1}^{\ell} m_i \cdot y_i + \sum_{i=1}^{\tau} b_i \cdot x_i \Big) \bmod x_0 \,.$$

$\mathsf{SIBDGHV.Add}(\mathsf{pk}, c_1, c_2)$. Output $c \leftarrow (c_1 + c_2) \bmod x_0$.

$\mathsf{SIBDGHV.Convert}(\mathsf{pk}, c)$. Output $c' \leftarrow 2 \cdot \langle \sigma, \mathsf{BitDecomp}_\eta(\mathbf{c}) \rangle \bmod x_0$ where $\mathbf{c} = \big(\lfloor c \cdot z_i \rceil \bmod 2^\eta \big)_{1 \leqslant i \leqslant \Theta}$.

$\mathsf{SIBDGHV.Mult}(\mathsf{pk}, c_1, c_2)$. Output $c' \leftarrow \mathsf{SIBDGHV.Convert}(\mathsf{pk}, 2 \cdot c_1 \cdot c_2 \bmod x_0)$.

$\mathsf{SIBDGHV.Decrypt}(\mathsf{sk}, c)$. Output $m_j \leftarrow \big((2c) \bmod p_j \big) \bmod 2$ for $j = 1, \ldots, \ell$

The constraints of our scale-invariant multi-slot scheme are the same as in Section 8.2.5.

Remark 8.11. Here again this describes a batch *leveled* fully homomorphic encryption scheme, but can be turned into a (pure) batch fully homomorphic encryption scheme.

8.3.2 Semantic Security

Here again, we show that the semantic security of our scheme can be based on the following variant of the decisional ℓ-Error-Free Approximate-GCD problem introduced in Definition 7.9.

Definition 8.12. Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$.

The *decisional squared ℓ-Error-Free-AGCD with additional elements* problem is: For ℓ coprime 2^ν-rough η-bit integers $p_1, \ldots, p_\ell$ and a uniformly chosen 2^ν-rough $q_0 \in [0, 2^\gamma/\pi^2)$ coprime with π, where $\pi = p_1 \times \cdots \times p_\ell$, given $x_0 = q_0 \cdot \pi^2$, polynomially many samples $\{x_i\}_i$ from $\mathbb{Z}_{x_0}$, and elements $\{y_i\}_{i=1}^\ell$, where y_i is sampled from

$$\frac{p_i - 1}{2} \cdot \prod_{\substack{j=1 \\ j \neq i}}^{\ell} p_j^2 + D_\rho^{(\ell)}(p_1^2, \ldots, p_\ell^2, q_0) \,,$$

to distinguish whether the samples are distributed uniformly or whether they are distributed according to the ℓ-AGCD distribution $D_\rho^{(\ell)}(p_1^2, \ldots, p_\ell^2, q_0)$.

Once again, this problem with parameters $(\gamma, \eta, \nu, \rho)$ reduces to the decisional ℓ-EF-AGCD problem with parameters $(\gamma, 2\eta, \nu, \rho)$.

As in Section 7.2.3, we have the following (straightforward) lemma:

Lemma 8.13. *Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$. The decisional squared 1-EF-AGCD with additional elements is hard under the decisional squared EF-AGCD assumption with additional element.*

We also have the following interesting reduction:

Lemma 8.14. *Let $\gamma, \eta, \nu, \rho \in \mathbb{N}$. The decisional squared ℓ-EF-AGCD with additional elements with parameters $(\gamma, \eta, \nu, \rho)$ is hard under the decisional squared 1-EF-AGCD with additional element with parameters $(\gamma - (\ell-1)\eta, \eta, \nu, \rho)$ (and therefore by Lemma 8.13, under the decisional squared EF-AGCD assumption with additional element with parameters $(\gamma - (\ell-1)\eta, \eta, \nu, \rho)$).*

The proof of this Lemma is straightforwardly adaptable from the proof of Lemma 7.11 (the key ingredients being the associativity of the CRT and a standard hybrid argument).

Finally, the following theorem shows that our scheme, without the public parameters $\mathbf{z}$ and $\boldsymbol{\sigma}$, is semantically secure under the decisional squared ℓ-Error-Free-AGCD with additional elements assumption

Theorem 8.15. *The above scale-invariant multi-slot DGHV scheme without the parameters* $\mathbf{z}$, $\boldsymbol{\sigma}$ *is semantically secure under the* $(\gamma, \eta, \nu, \rho)$*-decisional squared Error-Free-AGCD with additional element assumption.*

Then the proof of Theorem 8.15 is exactly the proof of Theorem 7.23, using a straightforward adaptation of Lemma 7.24 and Lemma 8.14 to conclude.

8.4 Practical Implementation

In this section, we provide concrete parameters and timings for the homomorphic evaluation of AES of Chapter 10 with our scale-invariant multi-slot DGHV scheme. We use the following existing optimizations:

1. Public-key compression: the technique in Section 7.5 (introduced in [CNT12]) enables to compress the ciphertexts in the public-key from γ to roughly $\ell \cdot \eta^2$ bits. We do not explicit the description here but it is essentially similar to Section 7.5.

2. Ciphertext expansion [CNT12]: the technique consists in generating the z_i's with a special structure instead of pseudo-random. Let δ be a parameter to be specified later. One generates a random z with $\kappa + \delta \cdot \Theta \cdot \eta$ bits of precision after the binary point, and one defines the z_i's for $\ell + 1 \leqslant i \leqslant \Theta$ as
$$z_i = \left[z \cdot 2^{i \cdot \delta \cdot \eta}\right]_{2^\eta},$$
keeping only κ bits of precision after the binary point for each z_i as previously. We fix $z_1, \ldots, z_\ell$ so that the previous equalities hold. Then the ciphertext expansion can be computed as follows, for all $\ell + 1 \leqslant i \leqslant \Theta$:
$$c_i = \lfloor c \cdot z_i \rceil \bmod 2^\eta = \lfloor c \cdot z \cdot 2^{i \cdot \delta \cdot \eta} \rceil \bmod 2^\eta.$$
Therefore computing all the z_i's (except the first ℓ) is now essentially a single multiplication $c \cdot z$. A lattice attack against this optimization is described in [CNT12]; the authors show that the attack is thwarted by selecting δ such that $\delta \cdot \Theta \cdot \eta \geqslant 3\gamma$.

8.4.1 Optimization of Scalar Product

We describe an additional optimization for computing the scalar product $c' = 2\langle \boldsymbol{\sigma}, \mathbf{c}' \rangle$ computed in Convert, similar to the ciphertext expand optimization above. The vectors $\boldsymbol{\sigma}$ and $\mathbf{c}'$ have $\eta\Theta$ elements. We first divide the vectors $\boldsymbol{\sigma}$ and $\mathbf{c}'$ into subvectors of Θ elements, and we compute the scalar products of the subvectors separately. In the following for simplicity we keep the same notations and now assume that $\boldsymbol{\sigma}$ and $\mathbf{c}'$ have Θ elements each.

We generate the vector $\boldsymbol{\sigma} \in \mathbb{Z}^\Theta$ such that:
$$\sigma_i = \left\lfloor \sigma \cdot 2^{i \cdot \delta \cdot \eta} \right\rceil + v_i$$
for small public corrections $|v_i| \leqslant 2^{\eta \cdot \ell}$ for all $1 \leqslant i \leqslant \Theta$, where the large public random σ has $\delta\eta\Theta$ bits of precision after the binary point, and $\gamma + \delta\eta\Theta$ bits in total. Then
$$\begin{aligned} c' &= 2\langle \boldsymbol{\sigma}, \mathbf{c}' \rangle = 2\sum_{i=1}^{n} \left\lfloor \sigma \cdot 2^{i \cdot \delta \cdot \eta} \right\rceil \cdot c'_i + 2\langle \mathbf{v}, \mathbf{c}' \rangle = 2\sum_{i=1}^{n} \left(\sigma \cdot 2^{i\delta\eta} + u_i\right) c'_i + 2\langle \mathbf{v}, \mathbf{c}' \rangle \\ &= 2\sigma \cdot \left(\sum_{i=1}^{n} c'_i \cdot 2^{i\delta\eta}\right) + 2\langle \mathbf{v}, \mathbf{c}' \rangle + u = \left\lfloor 2\sigma \cdot \left(\sum_{i=1}^{n} c'_i \cdot 2^{i\delta\eta}\right) \right\rceil + 2\langle \mathbf{v}, \mathbf{c}' \rangle + u', \end{aligned}$$
where $|u_i| \leqslant 1/2$, $|u| \leqslant \Theta$, and $u' \in \mathbb{Z}$ is such that $|u'| \leqslant \Theta + 1$. Then the scalar product becomes essentially one multiplication and another scalar product but with much smaller entries v_i's instead of σ_i's.

Therefore with vectors $\boldsymbol{\sigma}$ and $\mathbf{c}'$ with $\eta\Theta$ elements each instead of Θ, the scalar product $2\langle \boldsymbol{\sigma}, \mathbf{c}' \rangle$ becomes essentially η multiplications and another scalar product but with much smaller entries v_i's instead of σ_i's. Note that the size of c' is now $\gamma + \Theta\delta\eta$ bits instead of γ; therefore one must increase κ by twice the same additive factor (to support multiplications of two such converted ciphertexts).

Finally we use the following straightforward optimization: instead of using BitDecomp and PowersofTwo with bits, we use words of size ω bits instead. This decreases the size of the vector $\boldsymbol{\sigma}$ by a factor ω, at the cost of increasing the resulting noise by roughly ω bits. In particular the scalar product $2\langle\boldsymbol{\sigma}, \mathbf{c}'\rangle$ then requires essentially $\lceil\eta/\omega\rceil$ multiplications and another scalar product but with smaller entries v_i's instead of σ_i's. In our code we used $\omega = 64$.

8.4.2 Concrete Parameters and Benchmarking

In Section 8.2.5, we provided strict theoretical upper bounds on the noise growth during homomorphic operations to ensure correctness with *overwhelming probability*. In practice however, one expects a smaller noise growth on average and one could choose smaller bounds ensuring correctness with high probability only. This yields a huge gain in performance (allowing to reduce η, and thus γ) while still ensuring correctness most of the time. Therefore, for optimal performances in practice, one should select a parameter η as small as possible while still ensuring correctness with high probability. A similar approach as in Section 7.6.1 was used to derive practical parameters for our scale-invariant scheme (Table 8.1), parameters we will use for the homomorphic AES evaluation in Chapter 10.

Table 8.1 – Concrete Parameters for SIBDGHV.

Instance	λ	ℓ	ρ	η	$\gamma \times 10^{-6}$	τ, Θ	pk size
Toy	42	9	42	971	0.27	135	3.2 MB
Small	52	35	52	976	1.1	525	45 MB
Medium	62	140	62	981	4.2	2100	704 MB
Large	72	569	72	986	15.8	8535	11 GB
Extra	80	1875	86	993	35.9	28125	100 GB

We implemented the scale-invariant multi-slot DGHV scheme with a compressed public key (as in Section 7.5) in C++, using the GMP library. We provide benchmarks in Table 8.2.

Table 8.2 – Benchmarking for our Scale-Invariant Batch DGHV scheme with a compressed public key on an Intel Xeon E5-2690 at 2.9 GHz.

Instance	λ	ℓ	Keygen	Encrypt	Decrypt	Mult	Convert
Toy	42	9	0.5s	0.0s	0.0s	0.0s	0.1s
Small	52	35	11s	0.2s	0.0s	0.0s	0.3s
Medium	62	140	5min	3s	0.2s	0.0s	2.8s
Large	72	569	2h 50min	45s	3.3s	0.1s	33s
Extra	80	1875	213h	5min	24s	0.3s	277s

8.5 Conclusion

In this chapter, we proposed an adaptation of the scale-invariance technique introduced by Brakerski [Bra12] to the one-slot and multi-slot schemes over the integers introduced in Chapter 7. This technique allows to work with the same modulus throughout the whole evaluation process instead of considering a layer of moduli when using modulus switching (as described in [CNT12] for the one-slot DGHV scheme); and the noise growth is linear in the multiplicative depth of the circuit being evaluated.

We implemented our schemes in C++ using GMP, and described two aggressive optimizations to make our implementation more efficient. As a result, we could select parameters claiming to ensure 80 bits of security (contrary to Chapter 7). We selected parameters so as to evaluate the 60-level AES circuit in Chapter 10, and obtained timings for 80 bits of security comparable to the 72-bit secure parameters of Chapter 7 that could only evaluate circuits of depth 16 (the depth of the squashed decryption circuit plus one). As a consequence, this new technique yields exciting results that will be confirmed in Chapter 10.

Chapter 9

Minimal Number of Bootstrappings in Homomorphic Circuits

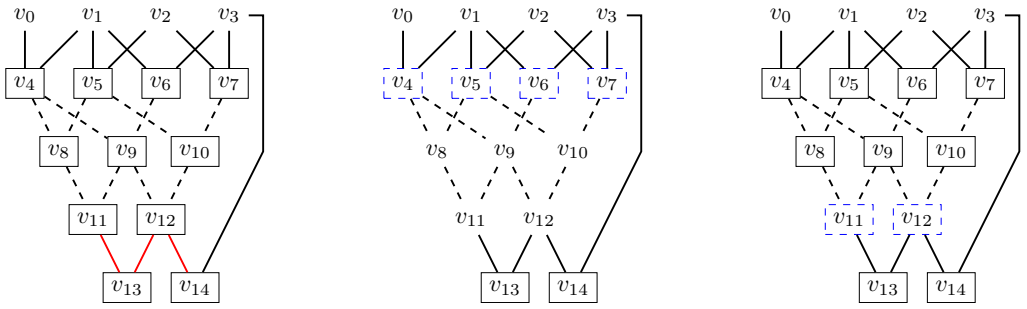
9.1 Introduction

We propose a method to compute the *exact* minimal number of bootstrappings required to homomorphically evaluate any circuit. Given a circuit (typically over $\mathbb{F}_2$ although our method readily extends to circuits over any ring), the maximal noise level supported by the considered fully homomorphic encryption (FHE) scheme and the desired noise level of circuit inputs and outputs, our algorithms return a minimal subset of circuit variables such that bootstrapping these variables is enough to perform an evaluation of the whole circuit. We introduce a specific algorithm for 2-level encryption (first generation of FHE schemes [Gen09, vDGHV10, CMNT11, BV11a, CNT12], including the schemes of Chapter 7) and an extended algorithm for $\ell_{\max}$-level encryption with arbitrary $\ell_{\max} \geqslant 2$ to cope with more recent FHE schemes (namely [BGV12, FV12, LTV12, BLLN13], and the schemes of Chapter 8). We successfully applied our method to a range of real-world circuits that perform various operations over plaintext bits. Practical results show that some of these circuits benefit from significant improvements over the naive evaluation method where all multiplication outputs are bootstrapped. In particular, we report that a circuit for the AES S-box put forward by Boyar, Matthews and Peralta [BMP13] admits a solution in 17 bootstrappings instead of 32. This will lead in Chapter 10 to a 88% faster homomorphic evaluation of AES using the batch DGHV scheme introduced in Chapter 7.

This chapter consists of the article On the Minimal Number of Bootstrappings in Homomorphic Circuits [LP13], cosigned with P. Paillier, and published at WAHC'13, the first Workshop on Applied Homomorphic Cryptography (held in conjunction with the 17th International Conference on Financial Cryptology and Data Security, FC 2013) [ABS13].

Noise Levels. In all known FHE schemes, a ciphertext c_i contains a noise r_i which grows along with homomorphic multiplications and decryption is ensured as long as r_i does not exceed a given bound, *i.e.* $r_i < r_{\max}$. Without loss of generality, we can assume that the noise is lower-bounded by the noise after a bootstrapping operation.[1] We adopt a simplified approach by associating with each ciphertext c_i a *discretized noise level* $\ell_i = 1, 2, \ldots$, where 1 is the noise level of ciphertexts resulting from a bootstrapping operation. Let c_1 (resp. c_2) be a ciphertext with noise level ℓ_1 (resp. ℓ_2). Gentry-like FHE schemes [Gen09, vDGHV10, BV11a, CMNT11, CNT12, CCK+13] (and in particular the scheme of Chapter 7) are such that $c_3 = c_1 + c_2$ has noise level $\ell_3 = \max(\ell_2, \ell_1)$ and $c_3 = c_1 \times c_2$ has noise level $\ell_3 = \ell_1 + \ell_2$, where $+$ and $\times$ respectively denote homomorphic addition and multiplication. Therefore in these schemes, the noise level grows exponentially with the number of homomorphic multiplications: to evaluate a circuit with L sequential layers of multiplications, one must impose the maximum noise level $\ell_{\max}$ to be larger than 2^L. This is practically unacceptable

[1] Note that in most FHE schemes, freshly generated ciphertexts have a smaller noise than the noise obtained after a bootstrapping operation, allowing the circuit evaluator to save several bootstrappings at the beginning of the circuit. However, it is possible that in *real-world* applications, data to be evaluated homomorphically will have been pre-processed and will not contain the smallest possible noise anymore.



(a) With no bootstrapping

(b) Bootstrapping after each multiplication

(c) Minimal solution

Figure 9.1 – Different bootstrapping solutions in a FHE scheme with $\ell_{\max} = 2$. Plain lines represent homomorphic multiplications while dashed lines represent homomorphic additions. The red lines in (a) reveal that the ciphertext noise will exceed the noise limit. Variables in a plain rectangle have a "large" noise ($\ell_i = \ell_{\max} = 2$) and the ones in a dashed blue rectangle are bootstrapped *i.e.* are re-encrypted to convert a "large" noise ($\ell_i = 2$) into a "small" noise ($\ell_i = 1$).

even for small values of L and one must resort to bootstrapping periodically as the circuit is being evaluated.

Note that our definition of noise levels neglects the logarithmic increase of the noise size after a homomorphic addition. This approximation is often considered in the literature and remains valid as long as the proportion of additions does not become overwhelming in the circuit. Clearly, our simplified model would become invalid outside of this context.

FHE schemes using Modulus-Switching. In FHE schemes using the modulus switching technique introduced in [BGV12], homomorphic addition still outputs ciphertexts of level $\ell_3 = \max(\ell_1, \ell_2)$. However, a homomorphic multiplication $c_3 = c_1 \times c_2$ now results in a noise level $\ell_3 = \max(\ell_1, \ell_2) + 1$. Thus, to evaluate a circuit with L layers of multiplications, one only requires $\ell_{\max} \geqslant L$. However, bootstrapping remains a cornerstone to achieve fully homomorphic encryption. When the depth of the circuit is not known at key generation time, bootstrapping is still required to evaluate a large circuit.

Minimizing bootstrappings. Overall, both one-modulus and scale-invariant FHE schemes must resort to bootstrapping in homomorphic circuit evaluation, either periodically or once in a while. However, the bootstrapping operation is reported as being the most drastic computational bottleneck in all known FHE implementations [GH11b, CMNT11, PBS11a, CNT12, CCK+13]. Worse, most of them merely perform a bootstrapping operation right after each multiplication, as suggested in [Gen09, vDGHV10]. It is easily seen though, as shown by the toy example depicted on Fig. 9.1, that this simple approach is often not optimal and that *fewer* bootstrappings may be sufficient to evaluate the whole circuit if positioned more judiciously.

Note that, even though finding a minimal solution is trivial and easily done by hand in Fig. 9.1, this optimization problem seems to become far more difficult with (even slightly) more complex circuits. Automated tools are therefore necessary to identify (one of) the smallest possible set of circuit variables whose bootstrapping will ensure a complete circuit evaluation in minimal time.

Our Contributions. We propose two efficient algorithms that automatically find an *exact* minimal solution for any given circuit, *i.e.* output a minimal list of circuit variables to which bootstrapping can be applied to evaluate the circuit. Section 9.2 introduces a first algorithm specific to the case of FHE schemes with a maximum noise level set to $\ell_{\max} = 2$. In Section 9.3, we extend our algorithm to support one-modulus FHE schemes handling up to $\ell_{\max} \geqslant 3$ noise levels. We show that the same extended algorithm can also be used with leveled schemes via a problem reformulation. Finally, Section 9.4 reports a number of experimental results on a range of real-world circuits, namely the

benchmarking circuits for MPC and FHE proposed by Smart and Tillich [ST], as well as circuits implementing the AES S-box suggested by Boyar and Peralta [BP12, BMP13].

9.2 Homomorphic Schemes with 2 Noise Levels

In this section, we consider a FHE scheme that can only handle two levels of randomness in ciphertexts, *i.e.* level-1 ciphertexts can either be added (yielding a level-1 ciphertext) or multiplied (yielding a level-2 ciphertext); however only addition can be performed on ciphertexts with levels $(1,2),(2,1)$ or $(2,2)$ since the result of a multiplication would not be decryptable. As a result, the scheme can only handle a single multiplication after each bootstrapping operation. This framework was heavily considered [Gen09, vDGHV10, CMNT11, PBS11a, BV11a, CNT12, CCK+13] and some implementations are available [PBS11b, CT12].

9.2.1 Stating the Problem

Let $\mathsf{C} = \mathsf{C}(n_1, n_2)$ be a Boolean circuit made of AND, XOR and NOT gates which takes as input n_1 bits and outputs n_2 bits. We denote by $\mathsf{C}^\dagger$ the same circuit as C where gates are replaced with homomorphic additions and multiplications.[2] Feeding $\mathsf{C}^\dagger$ with n_1 encrypted bits (under the FHE scheme), it will then output n_2 encrypted bits corresponding to the outputs of C applied on the same input bits in the clear. We denote by $\mathsf{V} = \{v_i : 1 \leqslant i \leqslant n\}$ the set of all single-assignment variables (ciphertexts) used in $\mathsf{C}^\dagger$ where $v_1, \ldots, v_{n_1}$ are the input variables and $v_{n-n_2+1}, \ldots, v_n$ the output variables. Now we assign a noise level $\ell_i \in \{1, 2, \ldots\}$ to each v_i as follows: the noise levels $\ell_1, \ell_2, \ldots, \ell_{n_1} \in \{1, 2\}$ are already fixed by the input variables $v_1, \ldots, v_{n_1}$. Using the two rules $\ell_{i_3} = \max(\ell_{i_1}, \ell_{i_2})$ when $v_{i_3} = v_{i_1} + v_{i_2}$ and $\ell_{i_3} = \ell_{i_1} + \ell_{i_2}$ when $v_{i_3} = v_{i_1} \times v_{i_2}$, we let noise levels automatically propagate throughout the circuit down to some output levels $\ell_{n-n_2+1}, \ldots, \ell_n$. Note that the noise levels of intermediate and output variables are left totally unbounded during that initial propagation and may therefore exceed by far the maximum level $\ell_{\max} = 2$ supported by the FHE scheme, meaning that the corresponding variables are in fact not decryptable. However, bootstrapping some variable v_i resets ℓ_i to 1 and it is easily seen that bootstrapping all variables $v_1, \ldots, v_n$ makes them all decryptable again: we then say that $\mathsf{C}^\dagger$ is evaluable. What we are after is a minimal subset $I \subseteq \{1, n\}$ such that bootstrapping v_i for all $i \in I$ has the same effect.

A Boolean reformulation. To each $v_i \in \mathsf{V}$ is assigned a Boolean value $b_i \in \{\mathsf{True}, \mathsf{False}\}$ that tells whether v_i is to be bootstrapped or not when evaluating $\mathsf{C}^\dagger$. We also define a Boolean mapping $\mathsf{B}(v_i)$ such that

$$\mathsf{B}(v_i) = \mathsf{True} \quad \text{if and only if} \quad \ell_i = 1\,.$$

We see that if $v_{i_3} = v_{i_1} + v_{i_2}$ then

$$\mathsf{B}(v_{i_3}) = b_{i_3} \vee \big(\mathsf{B}(v_{i_1}) \wedge \mathsf{B}(v_{i_2})\big)\,. \tag{9.1}$$

This is because $\ell_{i_3} = 1$ only if $\ell_{i_1} = \ell_{i_2} = 1$ or, as an alternate case, ℓ_{i_3} equals 2 when v_{i_3} is computed but bootstrapping v_{i_3} afterwards resets ℓ_{i_3} to 1. Moreover, if $v_{i_3} = v_{i_1} \times v_{i_2}$ then

$$\mathsf{B}(v_{i_3}) = b_{i_3}\,. \tag{9.2}$$

Indeed as the result of a multiplication v_{i_3} has level $\ell_{i_3} = 2$. The only way to get $\ell_{i_3} = 1$ is therefore to bootstrap v_{i_3} after computing it. We also see that $\mathsf{B}(v_i)$ is already determined for input variables since for $i = 1, \ldots, n_1$,

$$\mathsf{B}(v_i) = \begin{cases} \mathsf{True} & \text{if } \ell_i = 1, \\ b_i & \text{if } \ell_i \neq 1. \end{cases} \tag{9.3}$$

Overall, we see that the Boolean predicate B can also be propagated (as a multivariate Boolean expression) across the circuit using the above rules (9.1)–(9.3). This operation can be done statically given the description of the circuit and will result in a list of formal Boolean expressions for $\mathsf{B}(v_1), \ldots, \mathsf{B}(v_n)$ that only involve the 'bootstrapping' variables $b_1, \ldots, b_n$.

[2] XOR and NOT gates correspond to homomorphic additions and AND gates to homomorphic multiplications.

We now capture the fact that $\mathsf{C}^\dagger$ is evaluatable or not as a Boolean predicate ϕ^2_{C}. In order to ascertain the correctness of all variables of $\mathsf{C}^\dagger$, one must just ensure that all variables entering a multiplication have noise level 1. Hence

$$\phi^2_{\mathsf{C}} = \bigwedge_{v_k = v_i \times v_j \in \mathsf{C}^\dagger} \big(\mathsf{B}(v_i) \wedge \mathsf{B}(v_j)\big)\,. \tag{9.4}$$

Obviously, ϕ^2_{C} is a predicate involving $b_1, \ldots, b_n$ (or a subset thereof) and can be computed once B has been propagated throughout the circuit. All in all, evaluating $\mathsf{C}^\dagger$ with a minimal number of bootstrappings is reformulated as a Boolean satisfiability problem: ϕ^2_{C} must be satisfied with a minimal number of variables $b_1, \ldots, b_n$ set to True.

DNF and monotone predicates. We observe that the Boolean predicate $\phi^2_{\mathsf{C}} = \phi^2_{\mathsf{C}}(b_1, \ldots, b_n)$ is monotone since no negated literal $\neg b_i$ appears in ϕ^2_{C}. A monotone predicate is trivially satisfiable by setting all its variables to True. What we want, however, is to satisfy ϕ^2_{C} with as few b_i's set to True as possible. An exact solution to our problem would be to represent ϕ^2_{C} in Disjunctive Normal Form (DNF), *i.e.* as an XOR of ANDs. Given a DNF representation of ϕ^2_{C}, it is easy to identify an AND involving a minimal number of variables, thus providing a minimal bootstrapping configuration for $\mathsf{C}^\dagger$. However, noting $\mu(\phi^2_{\mathsf{C}}) \in [1, n]$ this minimal number, even just deciding whether $\mu(\phi^2_{\mathsf{C}}) \leqslant t$ for some $t \in [1, n]$ is a priori intractable:

Theorem 9.1 ([GHM05], Th. 3.4)**.** *Let ϕ be an n-variate Boolean monotone predicate and $t \in [1, n]$. Let $\mu(\phi)$ be the size of its smallest prime implicant. Deciding whether $\mu(\phi) \leqslant t$ is NP-complete.*

We therefore circumvent this obstacle by adopting a heuristic approach and further validate its effectiveness experimentally as reported later in the chapter.

9.2.2 A Heuristic Solver

We observe that ϕ^2_{C} is computed in Equation (9.4) as an accumulated conjunction: thus when propagating B across $\mathsf{C}^\dagger$, we systematically put each $\mathsf{B}(v_i)$ in minimal Conjunctive Normal Form (min-CNF), *i.e.* as an AND of XORs with as few terms as possible. Obviously $\mathsf{B}(v_i)$ becomes more complex (involves more b_i's) as the variable v_i is taken deeper in the circuit. However, the complexity increase remains incremental from $\mathsf{B}(v_{i_1}), \mathsf{B}(v_{i_2})$ to $\mathsf{B}(v_{i_3})$ for $v_{i_3} = v_{i_1}\ \mathsf{op}\ v_{i_2}$ and computing the min-CNF of $\mathsf{B}(v_{i_3})$ given the min-CNF of $\mathsf{B}(v_{i_1}), \mathsf{B}(v_{i_2})$ therefore requires a moderate computational effort. ϕ^2_{C} is then aggregated along the way as a min-CNF of other min-CNFs, which is easy to program. Once we are done collecting parts and putting together the multivariate predicate ϕ^2_{C}, we apply heuristic transformations on its min-CNF until it becomes small enough to allow a conversion to DNF using a standard algorithm. A minimal bootstrapping configuration is then selected from one of the smallest conjunctive clauses in the resulting DNF.

We apply 3 independent transformations on the min-CNF of ϕ^2_{C}:

1. **Bootstrap required variables:** if $\phi^2_{\mathsf{C}} = (\cdots) \wedge b_i \wedge (\cdots)$ for some b_i then set $b_i = \mathsf{True}$ and repeat the operation until no longer applicable;

2. **Remove redundant variables:** a variable b_i is *redundant* w.r.t. a variable b_j if every occurrence of b_i in a clause of ϕ^2_{C} appears together with an occurrence of b_j (but the converse might not be true). In other words, any clause c containing b_i is of the form $c = (\cdots) \vee b_i \vee (\cdots) \vee b_j \vee (\cdots)$. Setting $b_i = \mathsf{True}$ would of course lead to $c = \mathsf{True}$ but this will only remove all such clauses c from ϕ^2_{C}, whereas setting $b_j = \mathsf{True}$ instead might induce additional simplifications in other clauses of ϕ^2_{C}. Therefore, we set $b_i = \mathsf{False}$, propagate simplifications in the CNF of ϕ^2_{C}, repeat the operation until no longer applicable and restart with Step 1;

3. **Maintain minimal CNF:** Eliminate any clause that is tautologically implied by another clause of ϕ^2_{C}; repeat the operation until no longer applicable and restart with Step 1.

In practice, these transformations are reasonably efficient and allow us to reduce the min-CNF of ϕ^2_{C} in such proportions that converting it to DNF afterwards is either immediate or unnecessary (depending on the circuit C, ϕ^2_{C} sometimes reduces to True by itself along the way, which terminates our algorithm). Therefore, even though our method is unproven, we validated its practical effectiveness. We refer to Sections 9.4 and 9.4.2 for experimental results.

Remark 9.2. Note that the transformations 2 and 3 are such that one will not exhaust all the bootstrapping configurations that satisfy ϕ^2_{C}. However the final configuration is guaranteed to be minimal: since all minimal configurations are equivalent with respect to performance, all one cares about is finding one such configuration.

Remark 9.3. Note that one might also want to ensure that some output variables v_{n-n_2+j} for $j \in J \subseteq [1, n_2]$ have noise level 1 instead of 2. Now, resolving ϕ^2_{C} and bootstrapping these output variables might not yield a minimal solution. To address this case, we simply accumulate the predicates $\mathsf{B}(v_{n-n_2+j})$ for $j \in J$ into ϕ^2_{C} and apply the exact same strategy as above.

9.3 Extension to FHE Schemes with Many Noise Levels

Assume we are now given a FHE scheme that can handle $\ell_{\max} \geqslant 2$ levels of noise. Let c_1, c_2 and c_3 be ciphertexts with noise levels ℓ_1, ℓ_2 and ℓ_3 respectively. As discussed earlier, there exists essentially two different formulas for ℓ_3 when $c_3 = c_1 \times c_2$:

- $\ell_3 = \ell_1 + \ell_2$: this corresponds to the settings of one-modulus schemes [Gen09, vDGHV10, CMNT11, BV11a, CNT12, CCK+13, FV12, BLLN13, CLT14a].[3] In these schemes, the modulus remains the same after a multiplication but the noise increase depends on the amount of initial noises in the input ciphertexts. At most $\log_2(\ell_{\max})$ layers of homomorphic multiplications can be evaluated before resorting to bootstrapping;

- $\ell_3 = \max(\ell_1, \ell_2) + 1$: this corresponds to the FHE schemes using modulus-switching found in [BGV12, CNT12, LTV12]. The noise grows negligibly after a homomorphic multiplication, but the modulus is modified after each multiplication (therefore the relative amount of noise increases). This technique is known as modulus switching, wherein $\ell_{\max}$ different moduli are used to evaluate $\ell_{\max}$ layers of homomorphic multiplications without bootstrapping. Moreover two ciphertexts can only be added or multiplied when they have exactly the same noise level so that their underlying rings become identical. In the following, we assume that the cost of modulus switching for a variable v_i, *i.e.* incrementing its noise level, is negligible compared to the cost of a bootstrapping operation.

We generalize the method of Section 9.2 to FHE schemes with $\ell_{\max} \geqslant 2$ noise levels: Section 9.3.1 focuses on a extended algorithm that works with exponential schemes, and we show in Section 9.3.2 how to slightly modify $\mathsf{C}^\dagger$ in order to reuse the very same algorithm as a black-box to address linear schemes.

We recall that our goal is to minimize the number of bootstrappings needed to homomorphically evaluate the circuit $\mathsf{C}^\dagger$ on input $(v_i, \ell_i)_{1 \leqslant i \leqslant n_1}$. As above, we associate to every circuit variable $v_i \in \mathsf{V}$ a Boolean variable $b_i \in \{\mathsf{True}, \mathsf{False}\}$ that tells whether v_i is to be bootstrapped or not. Again, we construct a Boolean predicate $\phi^{\ell_{\max}}_{\mathsf{C}}$ as a function of $b_1, \ldots, b_n, \ell_1, \ldots, \ell_{n_1}$ that tells whether $\mathsf{C}^\dagger$ is evaluatable. We then rely on our heuristic solver of Section 9.2 to issue a minimal set $I \subseteq [1, n]$ such that $b_i = \mathsf{True}$ for all $i \in I$ implies $\phi^{\ell_{\max}}_{\mathsf{C}} = \mathsf{True}$.

9.3.1 Extension to One-Modulus FHE Schemes

To any variable $v_i \in \mathsf{V}$, we now associate a vector $\mathsf{B}(v_i) = (\mathsf{B}_{i,1}, \ldots, \mathsf{B}_{i,\ell_{\max}-1})^t$ with $(\ell_{\max} - 1)$ Boolean coefficients such that $\ell_i = j$ if and only if $B_{i,j}$ is the first coefficient set to True as j ranges from 1 to $\ell_{\max} - 1$, and $\ell_i = \ell_{\max}$ if none of the coefficients is True. We make use of the Boolean vector $\mathsf{B}(v_i)$ to encode the noise level ℓ_i of v_i and propagate it throughout the circuit as we did

[3] Note that the noise increase is quite different between the Gentry-like schemes [Gen09, CMNT11, CNT12, CCK+13] and the scale-invariant schemes [Bra12, FV12, BLLN13, CLT14a], but this does not affect our high-level description.

with $\mathsf{B}(v_i)$ in the binary case $\ell_{\max} = 2$. Let us describe in more detail how $\mathsf{B}(v_i)$ evolves when being propagated across the circuit:

- for $1 \leqslant i \leqslant n_1$, *i.e.* for input variables, set

$$\mathsf{B}_{i,j} = \mathsf{False} \quad \text{for } j \neq \ell_i \qquad \text{and} \qquad \mathsf{B}_{i,\ell_i} = \mathsf{True} \quad \text{if } \ell_i < \ell_{\max}\,.$$

- when $v_k = v_i + v_j$, set

$$\mathsf{B}(v_k) = \begin{pmatrix} b_k \vee \big(\mathsf{B}_{i,1} \wedge \mathsf{B}_{j,1}\big) \\ \big(\mathsf{B}_{i,1} \wedge \mathsf{B}_{j,2}\big) \vee \big(\mathsf{B}_{j,1} \wedge \mathsf{B}_{i,2}\big) \vee \big(\mathsf{B}_{i,2} \wedge \mathsf{B}_{j,2}\big) \\ \vdots \end{pmatrix}, \tag{9.5}$$

  Indeed, $\ell_k = 1$ if and only if v_k is bootstrapped or $(\ell_i, \ell_j) = (1,1)$, otherwise $\ell_k = 2$ if $(\ell_i, \ell_j) \in \big\{(1,2),(2,1),(2,2)\big\}$, etc. All vector coefficients $\mathsf{B}_{k,3}, \ldots, \mathsf{B}_{k,\ell_{\max}-1}$ are formed in the same fashion.

- when $v_k = v_i \times v_j$, set

$$\mathsf{B}(v_k) = \begin{pmatrix} b_k \\ \mathsf{B}_{i,1} \wedge \mathsf{B}_{j,1} \\ \big(\mathsf{B}_{i,1} \wedge \mathsf{B}_{j,2}\big) \vee \big(\mathsf{B}_{j,1} \wedge \mathsf{B}_{i,2}\big) \\ \vdots \end{pmatrix}. \tag{9.6}$$

  This multiplication expresses the fact that $\ell_k = \ell_i + \ell_j$. Indeed, $\ell_k = 1$ if and only if v_k is bootstrapped, $\ell_k = 2$ if and only if $\ell_i = \ell_j = 1$, and so forth.

Remark 9.4. Before explaining how to construct the Boolean formula $\phi_{\mathsf{C}}^{\ell_{\max}}$, let us give a couple of remarks on our representation. First of all, this representation does not imply that $\big(\mathsf{B}_{i,j} = \mathsf{True}$ and $\mathsf{B}_{i,m} = \mathsf{False}$ for $m \neq j\big) \iff \ell_i = j$, but that

$$\big(\mathsf{B}_{i,j} = \mathsf{True} \text{ and } \mathsf{B}_{i,m} = \mathsf{False} \text{ for } 1 \leqslant m < j\big) \iff \ell_i = j\,.$$

This allows us to simplify the formulas for homomorphic addition and multiplication as we do not need to check whether $\mathsf{B}_{i,m} = \mathsf{False}$ for $m > \ell_i$ (see $\mathsf{B}_{k,2}$ in Equation (9.5) and $\mathsf{B}_{k,3}$ in Equation (9.6)). Secondly, when all the elements of $\mathsf{B}(v_i)$ are False, this means that v_i is at the maximum level of noise $\ell_i = \ell_{\max}$. Therefore this representation nicely generalizes the one of Section 9.2.

We now construct the Boolean formula $\phi_{\mathsf{C}}^{\ell_{\max}}$ which tells whether the circuit is evaluatable by setting

$$\phi_{\mathsf{C}}^{\ell_{\max}} = \bigwedge_{v_i \in \mathsf{V}} (\ell_i \leqslant \ell_{\max}) = \bigwedge_{v_k = v_i \times v_j \in \mathsf{C}^\dagger} \Big(\bigvee_{1 \leqslant m \leqslant \ell_{\max}} \mathsf{B}_{k,m} \Big)\,.$$

Note that the clauses of $\phi_{\mathsf{C}}^{\ell_{\max}}$ encode the fact that to properly evaluate a homomorphic operation $v_k = v_i \;\mathsf{op}\; v_j$, one must just have $\ell_k \leqslant \ell_{\max}$. This is automatically guaranteed by induction for all additions; expressed on all multiplications, this constraint precisely gives the above expression. As before, we use minimal CNF representation to propagate $\mathsf{B}(v_i)$ throughout the circuit and aggregate all the clauses of $\phi_{\mathsf{C}}^{\ell_{\max}}$ on the way. This results in a min-CNF for $\phi_{\mathsf{C}}^{\ell_{\max}}$ to which we apply the same 3 simplifying transformations. We finally convert the resulting predicate to DNF (if necessary) to identify a minimal configuration.

Remark 9.5. Note that one might want to ensure that (a subset of) the output variables have noise levels bounded by some $\ell \leqslant \ell_{\max}$. One then aggregates in $\phi_{\mathsf{C}}^{\ell_{\max}}$ the clauses $\bigvee_{i \leqslant \ell} \mathsf{B}_{n-n_2+j,i}$ for $j \in [1, n_2]$ before solving the system.

Table 9.1 – Minimal number of bootstrappings with level-1 inputs and outputs.

Circuit $\mathsf{C}^\dagger$	$\boldsymbol{\ell_{\max}}$	Number of hom. multiplications in $\mathsf{C}^\dagger$	Exact minimal number of bootstappings
Adder 32 bits [ST]	2	127	127
Adder 32 bits [ST]	4	127	64
Comparator 32 bits [ST]	2	150	146
Comparator 32 bits [ST]	4	150	74
DES (expanded key) [ST]	2	18175	18041
DES (expanded key) [ST]	4	18175	8997
AES S-box [BP12]	2	32	19
AES S-box [BP12]	4	32	12
AES S-box [BMP13]	2	32	17
AES S-box [BMP13]	4	32	12

9.3.2 Extension to FHE Schemes using Modulus Switching

In this section, we explain how to deal with the case where $\ell_3 = \max(\ell_1, \ell_2) + 1$ when $c_3 = c_1 \times c_2$. Instead of adapting the previous method, we apply it as a black box to a modified version of the homomorphic circuit $\mathsf{C}^\dagger$. The modified circuit will no longer be consistent with its specification but can be treated by our algorithm regardless. The key idea is to see that one can simulate the linear framework in the exponential framework by replacing every homomorphic multiplication $c_3 = c_1 \times c_2$ with a subcircuit $c_3 = (c_1 + c_2) \times c_{1,2}$ where $c_{1,2}$ is a fixed ciphertext with noise level $\ell_{1,2} = 1$. Indeed, we get

$$\ell_3 = \max(\ell_1, \ell_2) + \ell_{1,2} = \max(\ell_1, \ell_2) + 1\,,$$

which is the wanted value in linear schemes. As mentioned, the correctness of the modified circuit as a homomorphic version of C is destroyed, but our extended algorithm remains applicable to it and will compute a minimal bootstrapping configuration in an oblivious fashion.

Note however that we need to slightly twitch the extended solver, otherwise solutions might suggest to bootstrap the newly introduced variables $v_{i,j}$. This would not make any sense as these variables have no real existence and only serve as helper variables in our simulation. We can easily circumvent this by not assigning a Boolean $b_{i,j}$ (or equivalently by forcing it to be False in $\mathsf{B}_{i,j}$) to the variables $v_{i,j}$. This eliminates the undesired collateral effect of seeing these variables being bootstrapped when solving $\phi_{\mathsf{C}}^{\ell_{\max}}$. We then successfully compute a minimal bootstrapping configuration from $\phi_{\mathsf{C}}^{\ell_{\max}}$ as previously described.

9.4 Practical Experiments

In this section, we discuss practical results obtained by applying our algorithms on several circuits (see Table 9.1). We implemented our basic and extended solvers using Mathematica 9 running on a 2.6 GHz Intel Core i7 with 16 GB of RAM. Although we did not specifically measure execution times, these range from a few seconds to a few hours depending on the circuit size and $\ell_{\max}$ (timings tend to grow exponentially with $\ell_{\max}$). We focused on the benchmarking circuits for MPC/FHE proposed by Smart and Tillich [ST], and on circuits put forward by Boyar and Peralta for the AES S-box [BP12, BMP13]. For each circuit, we computed the minimal number of bootstrappings needed to evaluate homomorphically that circuit with an exponential FHE scheme supporting $\ell_{\max} = 2$ or $\ell_{\max} = 4$ noise levels and with level-1 inputs and outputs, *i.e.*

$$\ell_1 = \cdots = \ell_{n_1} = 1 \qquad \text{and} \qquad \ell_{n-n_2+1} = \cdots = \ell_n = 1\,.$$

Table 9.1 reports the results we obtained by applying our algorithms to the selected circuits.

9.4.1 MPC/FHE Benchmark Circuits

Our results show that circuits given as reference by [ST] tend to be disappointing when $\ell_{\max} = 2$ as we find that the minimal number of bootstrapping required to evaluate them is nearly equal to the number of homomorphic multiplications, thus being very close to the (trivial) upper bound. This can be explained by the fact that these circuits are automatically generated from hardware components, and clearly not optimized: they were not constructed to be small in terms of gate count, or have a significantly smaller depth, etc. Their linear parts were not optimized either [BMP13]. Also note that setting $\ell_{\max} = 4$ instead of 2 divides the number of required bootstrappings by a factor nearly two.

9.4.2 The AES S-boxes of Boyar, Matthews and Peralta

To the best of our knowledge, the first 'real-life' circuit evaluated by a fully homomorphic encryption scheme is a circuit for AES encryption proposed by Gentry, Halevi and Smart [GHS12c]. However the authors are using modulus switching to get rid of the costly bootstrapping procedure by choosing a FHE scheme with $\ell_{\max} = 100$ so that the entire circuit can be evaluated at once. The drawback of this choice is that the public key becomes *prohibitively large* and required a server with 256GB of RAM to run the implementation and issue performance benchmarks. The authors suggested that bootstrapping might certainly be used as an optimization, *i.e.* as a way to balance the running time and the memory requirements.

In Chapter 10, we want to homomorphically evaluate the same AES encryption procedure, with the schemes introduces in Chapters 7 and 8. Since the former scheme is not a leveled homomorphic encryption scheme, one cannot set the number of levels too high and needs to use the bootstrap procedure repeatedly. Therefore, we will apply the technique describe in this chapter to minimize the number of bootstrapping for the homomorphic AES evaluation.

The non-linear part of AES, computing the S-box, cannot be performed by table lookups in an homomorphic implementation. We considered circuits for the AES S-box already optimized by Boyar, Matthews and Peralta with respect to gate count or depth [BMP13, BP12]. Our practical results are detailed on Table 9.1. Contrarily to the circuits of [ST], the latter circuits were optimized and we found that their minimal number of bootstrappings is nearly half the number of homomorphic multiplications when $\ell_{\max} = 2$. As a result, homomorphically evaluating an AES encryption with a 2-level FHE scheme can be boosted by a factor 1.88 by just choosing the circuit from [BMP13] and use our 17-bootstrapping optimal configuration

$$\{t_{21}, t_{22}, t_{23}, t_{24}, t_{26}, t_{29}, t_{33}, t_{36}, t_{40}, s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\} ,$$

as described in Chapter 10.

However, when $\ell_{\max}$ grows, the gain of minimal bootstrapping operations with respect to the case $\ell_{\max} = 2$ is smaller than for the circuits of [ST] (and even lower bounded by 8) due to the structure of these circuits.[4] Since the output variables are required to have a minimal noise level, the last reduction phase implies that the minimal solution consists in bootstrapping these output variables.

9.5 Conclusion

We introduced a method that computes the exact minimal number of bootstrappings required to homomorphically evaluate any circuit using any known FHE scheme. When $\ell_{\max} = 2$, the number of homomorphic multiplications is a strict upper bound on the minimal number of bootstrappings but significantly better figures can be found using our approach as exemplified by the circuit from [BMP13]. We see, however, that most commonly used circuits are disappointingly unoptimized with respect to their 'bootstrapping complexity'. As an avenue for future research, we suggest to explore algorithmic strategies to *build* bootstrapping-efficient circuits, *i.e.* to decrease their bootstrapping complexity by a specific design effort. Finally, it would be interesting to refine

[4] Note that these circuits are composed of three phases: top linear transformations, shared non-linear component, and bottom linear transformations.

our definition of noise levels to take into account the additional logarithmic effects induced by homomorphic operations, especially in the case of scale-invariant FHE schemes.

Chapter 10

Implementations of Homomorphic AES Evaluations

10.1 Introduction

This chapter investigates the concrete practicality of the schemes introduced in this part (Chapters 7 and 8) (and compares against existing results) by benchmarking the implementations against the 'standard' benchmark of implementing AES. We describe two homomorphic AES implementations, inspired from bitslicing techniques [Bih97, KS09] and using batching. The batching allowed to perform several independent AES decryption operations in parallel (or AES in counter mode).

This chapter consists of the implementations results described in the articles Batch Fully Homomorphic Encryption over the Integers [CCK+13], cosigned with J.H. Cheon, J.-S. Coron, J. Kim, M.S. Lee, T. Lepoint, M. Tibouchi and A. Yun, and published at Eurocrypt 2013 [JN13], and Scale-Invariant Fully Homomorphic Encryption over the Integers [CLT14a], cosigned with J.-S. Coron and M. Tibouchi, and published at PKC 2014 [Kra14]. The full versions of these papers are available at [CLT13a, CLT14b].

Sending Data to the Cloud. In typical real-world scenarios for using FHE with cloud applications, one or more clients communicate with a cloud service. They upload data encrypted with an FHE scheme under the public key of a specific user. The cloud can process this data homomorphically and return an encrypted result. Unfortunately, ciphertext expansion (*i.e.* the ciphertext size divided by the plaintext size) of current FHE schemes is prohibitive (thousands to millions). For example using techniques in [CNT12] (for 72 bits of claimed security), sending 4MB of data on which the cloud is allowed to operate, would require to send more than 73TB of encrypted data over the network. *Batching* several plaintexts into a single ciphertext [GHS12b, CCK+13, CLT14b] can improve on the required bandwidth; using [CCK+13] (*i.e.* the multi-slot scheme of Chapter 7) for example, the network communication would be lowered to around 280GB. However, this is still completely impractical.

To solve this issue, it was proposed in [NLV11] to instead send the data encrypted *with a block cipher* (in particular AES). The cloud service then encrypts the ciphertexts with the FHE scheme and the user's public key and *homomorphically decrypts* them before they are processed. Therefore, network communication is lowered to the data size (which is optimal) plus a costly *one-time setup* that consists of sending the FHE public key and an FHE encryption of the block cipher secret key (*cf.* Figure 10.1).

This suggestion requires a homomorphic evaluation of the block cipher decryption, which was successfully implemented for AES in [GHS12c] based on the BGV scheme [BGV12]. The resulting homomorphic evaluation of AES took 65 hours (on a Intel Xeon CPU running at 2.0 GHz) and processed 720 blocks in parallel (that is a relative time of 5 minutes per block). The AES circuit was chosen as a standard circuit to evaluate because it is nontrivial (but still reasonably small) and has an algebraic structure that works well with the plaintext space of certain homomorphic encryption schemes [GHS12c].

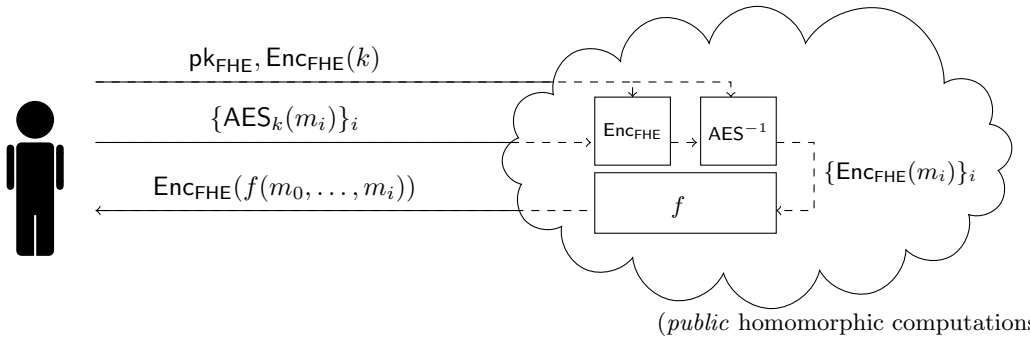


Figure 10.1 – Optimized communication with the cloud for homomorphic cryptography using AES.

Our Contributions. In this chapter, we use our multi-slot DGHV scheme (Chapter 7) and our scale-invariant multi-slot DGHV scheme (Chapter 8) to homomorphically evaluate the full AES encryption circuit.

First we describe two variants of homomorphic AES implementation, which we call *byte-wise bitslicing* and *state-wise bitslicing*. The first construction is similar to general-purpose bitslicing [Bih97, KS09], and represents the AES state by 8 ciphertexts $c_0, \ldots, c_7$, where the underlying plaintexts $\mathbf{m}_0, \ldots, \mathbf{m}_7$ encrypts bits of the AES state: $\mathbf{m}_0$ being the LSBs and $\mathbf{m}_7$ the MSBs. The second constructions completely splits the 128-bit AES state in 128 ciphertexts, where each ciphertext contains 1 bit of the AES state.

Next, we evaluate these implementations with the schemes described in Chapters 7 and 8. It appears that these schemes offer competitive performance for homomorphic cryptography: an amortized cost of about 23 seconds (resp. 3 minutes) per AES block at the 72-bit (resp. 80-bit) security level on a mid-range workstation. This is comparable to the timings presented by Gentry *et al.* at Crypto 2012 for their implementation of an RLWE-based scheme [GHS12c]. Note that our implementation using the multi-slot DGHV scheme uses bootstrapping, whereas the implementations of [GHS12c] and with our scale-invariant multi-slot DGHV scheme use leveled homomorphic encryption without bootstrapping.

While our implementations do not provide additional features nor significantly improved efficiency over the RLWE-based scheme of [GHS12c], we believe it is interesting to obtain FHE schemes with similar properties but based on different techniques and assumptions.

10.2 The AES Block Cipher

In this section, we briefly recall the Advanced Encryption Standard (AES) block cipher [FIP01]. AES uses a substitution-permutation network over blocks of 128 bits and can have three key lengths 128, 192 and 256 bits.

The AES state is represented as a 4×4 array of bytes, considered as elements of $\mathrm{GF}(2^8) \simeq \mathrm{GF}(2)[x]/(x^8 + x^4 + x^3 + x + 1)$ (filled column by column). Four operations are applied on the state:

AddRoundKey. This transformation perform a bitwise XOR between the state and a round key derived from the key schedule. We defer to [FIP01] for details on the key schedule of AES as we will not need it in this chapter.

SubBytes. This transformation is the only non-linear transformation of AES and substitutes each byte $\mathbf{b}$ of the state by the value $\mathbf{b} \mapsto \mathbf{M}\mathbf{b}^{254} + \mathbf{c}$, where the power function is performed over

$\mathrm{GF}(2^8)$,

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{c} = \mathsf{0x63}\,. \tag{10.1}$$

Note that the multiplication by $\mathbf{M}$ is viewed over $\mathrm{GF}(2)^8 \simeq \mathrm{GF}(2^8)$, where $\mathbf{b}$ is considered as its bit-representation vector.

MixColumns. This transformation consists in multiplying each column of the AES state (thus a vector of four bytes) by the matrix $\mathbf{M}'$ over $\mathrm{GF}(2^8)$, where

$$\mathbf{M}' = \begin{pmatrix} \mathsf{0x02} & \mathsf{0x03} & \mathsf{0x01} & \mathsf{0x01} \\ \mathsf{0x01} & \mathsf{0x02} & \mathsf{0x03} & \mathsf{0x01} \\ \mathsf{0x01} & \mathsf{0x01} & \mathsf{0x02} & \mathsf{0x03} \\ \mathsf{0x03} & \mathsf{0x01} & \mathsf{0x01} & \mathsf{0x02} \end{pmatrix}.$$

ShiftRows. This transformation cyclically shifts the last three rows of the state respectively by 1, 2 and 3 positions.

AES encryption consists of the successive operations AddRoundKey, $N_r - 1$ rounds (SubBytes, ShiftRows, MixColumns,AddRoundKey), SubBytes, ShiftRows and AddRoundKey, where $N_r = 10$ (resp. $N_r = 12$, resp. $N_r = 14$) for AES-128 (resp. AES-192, resp. AES-256).

10.3 Two Implementations of the Homomorphic AES

Throughout the rest of the chapter, we only consider AES-128 for the sake of simplicity; our methods are easily adaptable to AES-192 and AES-256. In this section we describe two implementations of homomorphic evaluation of an AES-128 circuit (HAES), using our multi-slot DGHV schemes of Chapters 7 and 8, with ℓ plaintext bits embedded in each ciphertext (*i.e.* a message space $\mathcal{M} = \mathbb{Z}_2^\ell$).

10.3.1 State-Wise Bitslicing

Recall that an AES state is constituted of 16 bytes, *i.e.* 128 bits. Let us represent the state of our HAES by 128 ciphertexts, where each ciphertext contains *one bit* of the AES state. We use the batching capability, *i.e.* the additional slots, to perform ℓ AES-128 encryptions *in parallel*. We call this representation *state-wise bitslicing*.

The state in our representation is composed of 128 ciphertexts $c_0, \ldots, c_{127}$, where the underlying plaintexts $\mathbf{m}_0, \ldots, \mathbf{m}_{127}$ are such that $\mathbf{m}_{i+j\cdot 8}[k]$ is the i-th bit of the j-th element of the state of the k-th AES.

AddRoundKey. From the 128-bit AES key, an expanded key is created with $(10+1)$ rounds subkeys (namely, one subkey at the beginning of the encryption, and one per round). Each round subkey is XORed at one point with the AES state. Thus, each bit of the subkey is XORed with the corresponding bit of the state.

The 128-bit AES subkeys are represented similarly to the HAES state, *i.e.* by 128 ciphertexts, one for each bit of the key. If the AES executed in parallel are with the same key, the key bit is put in each slot; otherwise the bit corresponding to the k-th AES is placed in the k-th slot.

Then the AddRoundKey stage simply consists of Add operations with the 128 ciphertexts of the encrypted AES key(s), one per ciphertext in the state. Therefore, this stage costs 128 Add operations.

ShiftRows. The permutation of the ShiftRows stage is applied on the indices of the ciphertexts of the HAES state. This stage is a relabeling of the indices of the ciphertexts of the HAES state, and therefore does not cost any homomorphic operation.

MixColumns. The MixColumns operation over the whole state can be viewed as

$$\begin{pmatrix} s'_0 & s'_4 & s'_8 & s'_{12} \\ s'_1 & s'_5 & s'_9 & s'_{13} \\ s'_2 & s'_6 & s'_{10} & s'_{14} \\ s'_3 & s'_7 & s'_{11} & s'_{15} \end{pmatrix} = \mathsf{0x02} \times \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix} \oplus \mathsf{0x03} \times \begin{pmatrix} s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \\ s_0 & s_4 & s_8 & s_{12} \end{pmatrix}$$
$$\oplus \begin{pmatrix} s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \\ s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \end{pmatrix} \oplus \begin{pmatrix} s_3 & s_7 & s_{11} & s_{15} \\ s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \end{pmatrix}.$$

In our implementation, we first store three copies of the HAES state (*i.e.* 3×128 ciphertexts) and we relabel their indices according to the previous operation. Next, we need to multiply by 0x02 the current HAES state and by 0x03 the first copy.

Algorithm 10.1 Multiplication by 0x02 in $\mathrm{GF}(2^8)$.

function MultiplyBy2($\mathbf{b} = \sum_{i=0}^{7} b_i x^i \in \mathrm{GF}(2)[x]$)
  $b'_0 \leftarrow b_7$
  $b'_1 \leftarrow b_0 \oplus b_7$
  $b'_2 \leftarrow b_1$
  $b'_3 \leftarrow b_2 \oplus b_7$
  $b'_4 \leftarrow b_3 \oplus b_7$
  $b'_5 \leftarrow b_4$
  $b'_6 \leftarrow b_5$
  $b'_7 \leftarrow b_6$
  return $\sum_{i=0}^{7} b'_i x^i$ ▷ $\mathbf{b} \cdot$ 0x02 over $\mathrm{GF}(2)[x]/(x^8 + x^4 + x^3 + x + 1)$
end function

Algorithm 10.2 Multiplication by 0x03 in $\mathrm{GF}(2^8)$.

function MultiplyBy3($\mathbf{b} = \sum_{i=0}^{7} b_i x^i \in \mathrm{GF}(2)[x]$)
  return MultiplyBy2($\mathbf{b}$) + $\mathbf{b}$ ▷ $\mathbf{b} \cdot$ 0x03 = $\mathbf{b} \cdot$ 0x02 + $\mathbf{b}$
end function

Multiplication of a byte $\mathbf{b}$ by 0x02 (resp. 0x03) over $\mathrm{GF}(2^8)$ is easy; see Algorithm 10.1 (resp. Algorithm 10.2).

When applying these algorithms on each block of 8 ciphertexts of the HAES state

$$(c_{0+8\cdot j}, \ldots, c_{7+8\cdot j}), \qquad j = 0, \ldots, 16\,,$$

each byte of the AES state is homomorphically multiplied by 0x02 (or 0x03), and this operation is performed in parallel on the ℓ AES states automatically. Therefore, $[3 + (3 + 8)] \times 16 = 224$ Add operations are performed during this step.

Finally, we need to add the four copies (possibly rotated or multiplied) of the state to get the final HAES state, and this is performed in $3 \times 128 = 384$ Add.

SubBytes. Recall that the SubBytes stage consists for each byte $\mathbf{b}$ in applying the transformation $\mathbf{b} \mapsto \mathbf{M}\mathbf{b}^{254} + \mathbf{c}$, where $\mathbf{M}, \mathbf{c}$ are defined in Equation (10.1).

A possible way to implement this stage could be to use Rivain and Prouff's method [RP10] that computes $\mathbf{b}^{254}$ from $\mathbf{b}$ with 4 multiplications over $\mathrm{GF}(2^8)$ and several squarings. The squaring of $\mathbf{b} = \sum_{i=0}^{7} b_i x^i \in \mathrm{GF}(2)[x]/(x^8 + x^4 + x^3 + x + 1)$ can be done only with XORs and therefore

with homomorphic additions, and therefore does not cost anything. The multiplication over $\mathrm{GF}(2^8)$ can be done by a Cauchy product and a reduction, and therefore an homomorphic multiplication consumes one level of noise.

For BDGHV, the multi-slot DGHV scheme of Chapter 7, after each multiplication 128 recryptions would be needed to recover a state with a "small" noise (*i.e.* small enough so that the 128 ciphertexts can be multiplied once without prior recryption). Therefore, computing $\mathbf{b}^254$ would require $4 \times 128 = 512$ BDGHV.Recrypt operations, which is a lot. To minimize the number of recryptions, we used the 115 gates circuit of Boyar, Matthew and Peralta [BMP13] to compute the S-box as proposed in Chapter 9. When applying this circuit homomorphically on each block of 8 ciphertexts of the HAES state

$$(c_{0+8\cdot j}, \ldots, c_{7+8\cdot j}), \qquad j = 0, \ldots, 16\,,$$

it suffices to bootstrap $9 + 8 = 17$ variables (*i.e.* apply 17 BDGHV.Recrypt) instead of 32 to recover a HAES state with "small" noise. This yields a $\approx 88\%$ faster computation of the AES S-box. Therefore, this stage costs 512 BDGHV.Mult, 272 BDGHV.Recrypt and 1328 BDGHV.Add. Note that under our representation the S-box circuit is evaluated in parallel over the k AES blocks.

For the sake of comparison, we used the same circuit for the scale-invariant DGHV scheme. This circuit consumes 6 levels of noise and require 32 SIBDGHV.Convert operations on each block of 8 ciphertexts of the HAES state. Therefore, this stage costs 512 SIBDGHV.Mult, 512 SIBDGHV.Recrypt and 1328 SIBDGHV.Add.

Final Cost. The AES encryption process consists of 11 AddRoundKey stages, 10 SubBytes, 9 MixColumns and 10 ShiftRows. Therefore, the final cost for BDGHV is 20160 BDGHV.Add, 2720 BDGHV.Recrypt and 5120 BDGHV.Mult, and the final cost for SIBDGHV is 20160 SIBDGHV.Add, 5120 SIBDGHV.Recrypt and 5120 SIBDGHV.Mult. However, a fine management of the noise allows us to reduce the number of BDGHV.Recrypt to 2448 (namely, we do not need to bootstrap *at all* in SubBytes during the first round).

10.3.2 Byte-Wise Bitslicing

In this section, we propose a new representation that will use permutations over plaintext slots. We described how to permute plaintext slots for the BDGHV scheme, for free during the Recrypt procedure, in Section 7.4.3. This is made possible by including in the public key specifically designed encryptions of the permuted secret key bits.

As a consequence, we only consider the BDGHV scheme in this section.

Recall that the AES state is a matrix of 4×4 bytes. It can be viewed as a 16-byte vector when reading the bytes by column. We define a representation called *byte-wise bitslicing* in which the HAES state will be composed of 8 ciphertexts, each ciphertext containing one and exactly one bit *of each byte* of the AES state (this requires batching). This construction is similar to general-purpose bitslicing [Bih97, KS09]. We also use the batching capability to perform $\ell' = \lfloor \ell/16 \rfloor$ AES-128 encryptions in parallel.

The state in our representation is composed of 8 ciphertexts $c_0, \ldots, c_7$, where the underlying plaintexts $\mathbf{m}_0, \ldots, \mathbf{m}_7$ are such that $\mathbf{m}_i[k \cdot 16 + j]$ is the i-th bit of the j-th element of the state of the k-th AES (see Figure 10.2). Thus $\mathbf{m}_0$ represents the LSBs of the bytes of the AES states for the ℓ' AES plaintexts, and $\mathbf{m}_7$ the MSBs.

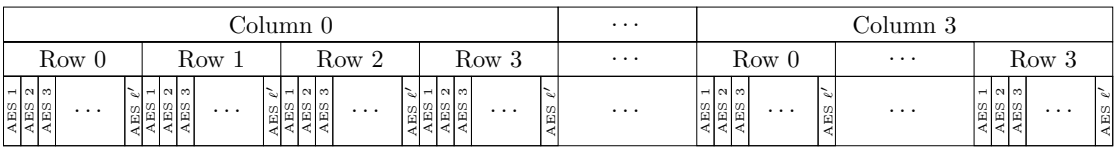


Figure 10.2 – Bit ordering in $\mathbf{m}_i$ in the byte-wise bitslicing representation.

AddRoundKey. Here again we construct the round subkeys with the same structure as the HAES state (*i.e.* 8 ciphertexts); note that we repeat each bit of the round subkey ℓ' times. Therefore, the AddRoundKey stage only consists in adding the corresponding ciphertexts with BDGHV.Add as the underlying operation is a XOR on the plaintext bits. This operation consists of 8 BDGHV.Add operations.

MixColumns. As in the previous state-wise bitslicing representation, we define three copies of the HAES state (*i.e.* 24 additional ciphertexts) and we rotate them according to the permutations ζ_1, ζ_2 or ζ_3, with

$$\zeta_i(I \times \ell' + K) = \zeta_{\mathrm{MC}_i}(I) \times \ell' + K, \qquad 0 \leqslant K \leqslant \ell' - 1, 0 \leqslant I \leqslant 15\,,$$

the permutation ζ_{MC_i} being defined as

$$\zeta_{\mathrm{MC}_i}(I) = \lfloor I/4 \rfloor \cdot 4 + (I + i \bmod 4), \qquad 0 \leqslant I \leqslant 15\,.$$

This costs $3 \times 8 = 24$ BDGHV.Recrypt.

Next, we need to multiply by 0x02 the current HAES state and by 0x03 the first copy and this is easy as previously thanks to Algorithms 10.1 and 10.2. When applying these algorithms on the HAES state $(c_0, \ldots, c_7)$ instead of $(b_0, \ldots, b_7)$, the multiplication by 0x02 and 0x03 are performed *in parallel* over all the bytes of an underlying AES state, *and* also on the ℓ' AES states. Therefore, $3 + (3 + 8) = 14$ BDGHV.Add operations are performed during this step.

Finally, we need to add the four copies (possibly rotated or multiplied) of the state to get the final HAES state, and this is performed in $3 \times 8 = 24$ BDGHV.Add operations.

SubBytes. As for the state-wise bitslicing variant, we used the 115 gates circuit of Boyar, Matthews and Peralta [BMP13] to compute the S-box with a small number of Recrypt. This stage costs 32 BDGHV.Mult, 17 BDGHV.Recrypt and 83 BDGHV.Add. Note that under our representation the S-box circuit is evaluated in parallel over the 16 bytes of an AES state, and also on the ℓ' AES blocks.

Note that our representation is very well adapted to this SubBytes stage. Indeed, since the same operation needs to be performed on *all* the bytes of the AES state, for the ℓ' AES blocks performed in parallel, manipulating the ciphertext c_0 as if it was the LSB of a byte of an AES state, and c_7 as the MSB, evaluating the previous circuit allows to perform the SubBytes stage in *parallel* not only over the 16 bytes of an AES state, but also on the ℓ' AES blocks!

ShiftRows. Contrary to the state-wise bitslicing representation, the ShiftRows stage is not only a reordering of the indices of the ciphertext. On the contrary, it consists in performing the permutation ζ_{SR} on the bytes of the AES state, where the Cauchy's two-line notation of the permutation is

$$\zeta_{\mathrm{SR}} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 13 & 10 & 7 & 4 & 1 & 14 & 11 & 8 & 5 & 2 & 15 & 12 & 9 & 6 & 3 \end{pmatrix}$$

Since we "sliced" the bytes of the AES state in our representation, we will need to apply a similar permutation on each ciphertext of the state. Since we are performing ℓ' AES blocks in parallel, we need to consider the permutation ζ defined by

$$\zeta(I \times \ell' + K) = \zeta_{\mathrm{SR}}(I) \times \ell' + K, \qquad 0 \leqslant K \leqslant \ell' - 1, 0 \leqslant I \leqslant 15\,.$$

Next, we need to permute each of the c_i's of the HAES state by ζ to perform the ShiftRows stage. As mentioned in Section 7.4.3, rotating the slots is "for free" when performed during a Recrypt. Now, we perform a recryption on each element of the HAES state at the end of the SubBytes stage. Thus, instead of using the regular σ_i's as Section 7.4.2, we use the σ_i^{ζ}'s as in Section 7.4.3 and the ShiftRows stage will be obtained at the end of the SubBytes stage at *no additional cost.*

Final Cost. The AES encryption process consists of 11 AddRoundKey stages, 10 SubBytes and ShiftRows stages and 9 MixColumns stages. Therefore, the final cost is 1260 BDGHV.Add, 386 BDGHV.Recrypt and 320 BDGHV.Mult. As before, a fine management of the noise allows us to reduce the number of BDGHV.Recrypt to 377.

10.4 Implementation Results

We implemented proof-of-concept implementations of a homomorphic encryption of AES with the test vector of [FIP01, Appendix B], using the C++ implementations of BDGHV and SIBDGHV mentioned in Chapters 7 and 8. We provide our results in Tables 10.1.

In particular, we show that our most efficient homomorphic AES evaluation takes just over 4 days, but can process 1875 blocks in each evaluation, yielding an amortized rate of just over three minutes per block. This is comparable to the five minutes per AES block of the homomorphic AES evaluation using the BGV scheme [BGV12] in [GHS12c].

10.4.1 Some Thoughts about Homomorphic Evaluations

Latency versus Throughput. Let us define the two notions latency and throughput associated to a homomorphic evaluation. We say that the *latency* of a homomorphic evaluation is the time required to perform the entire homomorphic evaluation. Its *throughput* is the number of blocks processed per unit of time.

The results presented in Table 10.1 illustrate the fact that a careful design of the algorithm to be evaluated can decrease the latency. Namely, to have a latency as small as possible, the byte-wise bitslicing representation is more adapted than the state-wise bitslicing representation (18 hours versus more than 100 hours).

The state-wise bitslicing representation (and the representations in [GHS12c]) were chosen to maximize the throughput, by allowing more blocks to be processed at once. Therefore, we can claim a "small" *relative time per AES block* while the latency is several dozens of hours. However, "real world" homomorphic evaluations (likely to be used in the cloud) should be implemented in a transparent and user-friendly way. It is therefore questionable whether maximizing the throughput by treating a lot of blocks in parallel is: (1) suitable for further processing of data, (2) worth the impact on the latency. In particular, it might only be interesting when this processing is identical over each block (which is likely *not* to be the case in real world scenarios). Overall, one should rather select parameters to have the latency as small as possible. The throughput can be increased by running the homomorphic evaluations in a cluster.

Cloud Computations. The purpose of the scenario in which the data is sent encrypted with a block cipher to the cloud is that, once the data arrives in the cloud and has been homomorphically decrypted, the cloud can perform *more* homomorphic operations on it. With a scheme that implements bootstrapping, *e.g.* the BDGHV scheme of Chapter 7, there is no restriction to that. But in practice, the homomorphic encryption schemes often are not implemented with bootstrapping. In particular, we did not implement the bootstrapping in the SIBDGHV scheme, nor was it implemented in [GHS12c]. For the AES evaluations the parameters of the leveled homomorphic scheme were chosen so that it can homomorphically evaluate the AES decryption without bootstrapping, but not much more. Taking into account a certain amount of computations after the homomorphic decryption either requires larger parameters to ensure correctness, or the implementation of bootstrapping. Following the former approach, it should be noted that parameter selection needs to be done ensuring correctness for a circuit including the block cipher operation *and* the desired application function. Overall, depending on the specific application, performance might become worse than indicated in the current results.

10.5 Conclusion

In this chapter, we proposed two representations to perform a homomorphic AES evaluation. These representations use several slots (*i.e.* the batching capability) to process several AES states in parallel. Then we homomorphically evaluated AES with the multi-slot FHE schemes designed in Chapters 7 and 8. Our proof-of-concept implementations in C++, using the big integer library GMP, yield that AES can be evaluated *completely* in about 18 hours, or when processing several blocks in parallel, in about 3 minutes per block on a mid-range computer. These results are of the same order of efficiency than the homomorphic AES evaluation of Gentry, Halevi and Smart [GHS12c] based on the BGV fully homomorphic encryption scheme [BGV12].

Table 10.1 – Benchmarking of homomorphic AES encryptions using BDGHV and SIBDGHV.

(a) Timings for byte-wise representation using BDGHV, on a desktop computer (Intel Core i7 at 3.4Ghz, 32GB RAM).

Instance	λ	ℓ	# of enc. in parallel	AddRoundKey	ShiftRows and SubBytes	MixColumns	**Total AES** (in hours)	**Relative time**
Toy	42	16	1	0.006s	2.2s	3s	**0.013**	**48s**
Small	52	48	3	0.04s	21s	29s	**0.125**	**2min 30s**
Medium	62	144	9	0.3s	210s	290s	**1.25**	**8min 20s**
Large	72	528	33	1.6s	2970s	4165s	**18.3**	**33min**

(b) Timings for state-wise representation using BDGHV, on a desktop computer (Intel Core i7 at 3.4Ghz, 32GB RAM).

Instance	λ	ℓ	# of enc. in parallel	AddRoundKey	SubBytes	ShiftRows	MixColumns	**Total AES** (in hours)	**Relative time**
Toy	42	10	10	0.06s	33s	0s	0.02s	**0.08**	**29s**
Small	52	37	37	0.06s	309s	0s	0.09s	**0.74**	**1min 12s**
Medium	62	138	138	4.5s	3299s	0s	0.44s	**7.86**	**3min 25s**
Large	72	531	531	27s	47656s	0.04s	2.8s	**113**	**12min 46s**

(c) Timings for state-wise representation using SIBDGHV, on an Intel Xeon E5-2690 at 2.9 GHz.

Instance	λ	ℓ	# of enc. in parallel	AddRoundKey	SubBytes	ShiftRows	MixColumns	**Total AES** (in hours)	**Relative time**
Toy	42	9	9	0.0s	1.5s	0.0s	0.0s	**0.004**	**1.7s**
Small	52	35	35	0.1s	9.9s	0.0s	0.0s	**0.027**	**2.9s**
Medium	62	140	140	0.3s	80.5s	0.0s	0.1s	**0.22**	**5.8s**
Large	72	569	569	2.1s	21min	0.0s	0.6s	**3.58**	**23s**
Extra	80	1875	1875	6.9s	10h 9min	0.1s	1.6s	**102**	**195s**

As a consequence, it appears that our scale-invariant multi-slot DGHV scheme offers a competitive alternative to the lattice-based leveled homomorphic encryption schemes, while relying on a different hardness assumption.

The AES circuit was chosen as a standard circuit to evaluate because it is nontrivial (but still reasonably small) and has an algebraic structure that works well with the plaintext space of certain homomorphic encryption schemes [GHS12c]. However, there might be other ciphers that are more suitable for being evaluated under homomorphic encryption, and this remains a promising research area.

For example, we propose with Michael Naehrig in [LN14a] to consider lightweight block ciphers instead of AES. Indeed, these block ciphers were engineered to be extremely small, easy to implement and efficient in hardware. Now due to the limitations of current homomorphic encryption schemes, this hardware optimized structure yields likely candidates for homomorphic cryptography. In [LN14a], we present an homomorphic evaluation of the block cipher Simon [BSS+13] unveiled in June 2013 by the U.S. National Security Agency. Using the SIBDGHV scheme of Chapter 8, we show that a full-fledged homomorphic Simon-32/64 evaluation (having 64 bits of security) can be evaluated in about 10 minutes, for an amortized time of *3 seconds per block*. Our main contributions however are focusing on the lattice-based homomorphic encryptions schemes YASHE [BLLN13] and FV [FV12], which finally appear to be more competitive than the integer-based schemes. Independently, Doröz, Shahverdi, Eisenbarth and Sunar proposed in [DSES14] to homomorphically evaluate the lightweight block cipher Prince [BCG+12], and obtained very promising performances; in particular Prince is *particularly* adapted to FHE schemes. Designing a FHE-friendly block cipher is a very mainstream research subject and could accelerate the use of FHE schemes in "real world" applications.

Part Three

Design and Implementation of Multilinear Maps over the Integers

Overview

In 2013, Joux, Boneh and Franklin received the prestigious Gödel Prize for establishing the field of pairing-based cryptography in the early 2000's [Jou00, SOK00, BF01]. They used *bilinear* maps to provide compelling new and rich applications in cryptography for which no other efficient implementation is known. The impact of their research is tremendous, and applications of bilinear maps have become too numerous to name, but are often at the core of the latest advances in cryptography. In particular, they are currently being investigated to bring secure authentication and *privacy* to the end-users.

A couple of years after the introduction of pairing-based cryptography, Boneh and Silverberg provided evidence that cryptographic multilinear maps (*i.e.* generalizations of bilinear maps) were likely to have astounding applications in cryptography, even though constructing such a cryptographic primitive remained a challenging open problem [BS03]. Several subsequent works based on this virtual primitive gave birth to new applications [RS09, PTT10, Rot13], but in the absence of concrete construction this research area was not tremendously active.

Everything changed in 2013 when Garg, Gentry and Halevi presented in a breakthrough result a candidate multilinear maps scheme based on ideal lattices [GGH13a]. Even though their scheme differs in a number of ways compared to the "ideal" virtual primitive introduced by Boneh and Silverberg, they showed that their approximation is good enough for a number of applications. This powerful new cryptographic tool has a tremendous impact in theoretical cryptography and opened a floodgate of exciting developments in the last months.

Chiefly among them is a candidate construction for general purpose *program obfuscation* proposed by Garg, Gentry, Halevi, Raykova, Sahai and Waters in 2013 [GGH$^+$13b]. Namely, given any two circuits of the same size that compute the same functionality, they proposed an obfuscation construction such that no polynomial time adversary can distinguish between the obfuscation of the first circuit with respect to the obfuscation of the second circuit. Goldwasser and Rothblum gave a strong philosophical justification for indistinguishability obfuscation [GR07], called *best possible obfuscation*, as such an obfuscator guarantees that its output hides as much as the input circuit as possible. Once again, this breakthrough result brought many exciting new applications and fascinating new foundational problems for the field to study. Similarly to fully homomorphic encryption, the Holy Grail of cryptography [Mic10], to our surprise, indistinguishability obfuscation appears to suffice to construct many other cryptographic applications and has an exciting impact on the field. Among others, it allows to construct deniable encryption [SW13], round optimal multiparty secure computation [GGHR14], multiparty key exchange, efficient traitor tracing, broadcast encryption with optimal ciphertext size [BZ13], to remove the random oracle [HSW13b] and many others things...

Numerous other applications of multilinear maps were (and still are) discovered, such as identity-based key exchange, broadcast encryption system with optimal ciphertext size, policy-based key distribution [BW13], removing random oracles [FHPS13, HSW13a], verifier-based pass-

word-authenticated key exchange [BP13], forward secure non-interactive key exchange [PS14], attribute-based encryption for circuits [GGH+13c], and certainly many others...

In this part, we will describe an other construction of approximate multilinear maps, based on the framework introduced in [GGH13a], based on different techniques and assumptions, that is conceptually simpler. We sustain our new scheme with a thorough analysis of the attacks and we describe the *first* implementation of approximate multilinear maps, thanks to a number of heuristic modifications to our scheme. Moreover, some hardness assumptions easy for the GGH scheme appear to hold for our scheme. Note that our scheme offers the same flexibility as the GGH scheme, and in particular all the aforementioned applications of multilinear maps are directly – and sometimes only – instantiable with our scheme.

Chapter 11

Multilinear Maps over the Integers

11.1 Introduction

Extending bilinear elliptic curve pairings to multilinear maps is a long-standing open problem. The first plausible construction that approximates cryptographic multilinear maps has been described by Garg, Gentry and Halevi at Eurocrypt 2013, based on ideal lattices.

In this chapter, we build upon their construction and describe an alternative construction of (approximate) multilinear maps that works over the integers instead of ideal lattices, similar to the DGHV fully homomorphic encryption scheme (*cf.* Part II). Our construction is not a mere adaptation of Garg *et al.* scheme over the integers. In particular, we describe a different technique for proving the full randomization of encodings: instead of Gaussian linear sums, we apply the classical Leftover Hash Lemma over a quotient lattice. Moreover, in contrast with Garg *et al.* scheme, multilinear analogues of useful, base group assumptions like DLIN appear to hold in our setting. Looking ahead, in Chapter 12, we will describe the first implementation of multilinear maps and show that our construction is arguably practical as we perform a 7-party (resp. 26-party) Diffie-Hellman key exchange in about 40 seconds (resp. 5 minutes) per party.

This chapter includes most of the article Practical Multilinear Maps over the Integers [CLT13b], cosigned with J.-S. Coron and M. Tibouchi, and published at Crypto 2013 [CG13a]. The full version of the article is available at [CLT13c].

Background. In 2003, Boneh and Silverberg [BS03] studied a generalization of cryptographic bilinear maps called cryptographic multilinear maps. However they were pessimistic about the existence of such maps from the realm of algebraic geometry, and the construction of such a primitive remains a challenging open problem. In 2013, Garg, Gentry and Halevi presented in a breakthrough result a candidate construction for (approximate) multilinear maps, based on ideal lattices. They introduced a new primitive called *graded encoding systems* (named after graded algebra) that differs from "ideal" multilinear maps in the sense that each encoding has a level, and multiplying two encodings adds their level. A κ-linear map now consists of multiplications, and an additional parameter is provided and allows to test, only at a given level κ, whether an encoding encodes $\mathbf{0}$ or not. This richer primitive (because leveled) has unfortunately the same drawbacks as fully homomorphic encryption: the encodings contain noise that increases after each multiplication, the size of the public parameters are polynomial in the maximum level[1] and this first candidate is incredibly unpractical.

Our Results and Techniques. Our main contribution is to describe a different and conceptually simpler construction of approximate multilinear maps that works over the integers instead of ideal lattices, similar to the DGHV fully homomorphic encryption scheme and its batch variant (*cf.* Chapter 7).

Our construction offers the same flexibility as the original from [GGH13a]; in particular it can be modified to support the analogue of asymmetric maps and composite-order maps. Moreover, it

[1] The dependence between the degree and the parameter-size prevents them (and will prevent us) from realizing applications such as the ones envisioned by [PTT10] because they need "compact" maps.

does not seem vulnerable to the "zeroizing" attack that breaks base group hardness assumptions like the analogues of DLIN and subgroup membership for the multilinear maps of [GGH13a]. Since those assumptions are believed necessary to adapt constructions of primitives like adaptively secure functional encryption, NIZK or verifier-based password-authenticated key exchange [BP13], our construction seems even more promising for applications than [GGH13a].

As in [GGH13a], the security of our construction relies on new assumptions; it cannot be derived from "classical" assumptions such as the Approximate-GCD assumption used in Chapter 7. We describe various possible attacks against our scheme; this enables us to derive parameters for which our scheme remains secure against these attacks.

Our new construction works as follows: one first generates n secret primes p_i and publishes $x_0 = \prod_{i=1}^{n} p_i$ (where n is large enough to ensure correctness and security); one also generates n primes g_i, and a random secret integer z modulo x_0. A level-k encoding of a vector $\mathbf{m} = (m_i) \in \mathbb{Z}^n$ is then an integer c such that for all $1 \leqslant i \leqslant n$:

$$c \equiv \frac{r_i \cdot g_i + m_i}{z^k} \pmod{p_i} \tag{11.1}$$

for some small random integers r_i; the integer c is therefore defined modulo x_0 by CRT. It is clear that such encodings can be both added and multiplied modulo x_0, as long as the numerators remain smaller than the p_i's. In particular the product of κ encodings c_j at level 1 gives an encoding at level κ where the corresponding vectors $\mathbf{m}_j$ are multiplied componentwise. For such level-κ encodings one defines a zero-testing parameter p_{zt} with:

$$p_{zt} = \sum_{i=1}^{n} h_i \cdot \left(z^\kappa \cdot g_i^{-1} \bmod p_i\right) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0$$

for some small integers h_i. Given a level-κ encoding c as in Equation (11.1), one can compute $\omega = p_{zt} \cdot c \bmod x_0$, which gives:

$$\omega = \sum_{i=1}^{n} h_i \cdot \left(r_i + m_i \cdot (g_i^{-1} \bmod p_i)\right) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0 \ .$$

Then if $m_i = 0$ for all i, since the r_i's and h_i's are small, we obtain that ω is small compared to x_0; this enables to test whether c is an encoding of $\mathbf{0}$ or not. Moreover for non-zero encodings the leading bits of ω only depend on the m_i's and not on the noise r_i; for level-κ encodings this enables to extract a function of the m_i's only, which eventually defines as in [GGH13a] a degree-κ (approximate) multilinear map.[2]

Our second contribution is to describe a different technique for proving the full randomization of encodings. As in [GGH13a] the randomization of encodings is obtained by adding a random subset-sum of encodings of $\mathbf{0}$ from the public parameters. However as in [GGH13a] the Leftover Hash Lemma (LHL) cannot be directly applied since the encodings live in some infinite ring instead of a finite group. The solution in [GGH13a] consists in using linear sums with Gaussian coefficients; it is shown in [AGHS13] that the resulting sum has a Gaussian distribution (over some lattice). As noted by the authors, this can be seen as a "Leftover Hash Lemma over lattices". In this paper we describe a different technique that does not use Gaussian coefficients; instead it consists in working modulo some lattice $L \subset \mathbb{Z}^n$ and applying the Leftover Hash Lemma over the quotient $\mathbb{Z}^n/L$, which is still a finite group. This technique was already used in [CCK+13, CLT13a] to prove the security of a batch extension of the DGHV scheme.[3] We provide here a more formal description of our "Leftover Hash Lemma over lattices". Note that our technique can independently be applied to the original encoding scheme from [GGH13a], while the Gaussian sum technique from [AGHS13] is also applicable to ours.[4]

In Chapter 12, we will describe the *first* implementation of cryptographic multilinear maps, provide concrete parameters and timings to do a N-partite Diffie-Hellman key exchange.

[2] Technically for p_{zt} we use a *vector* of integers instead of a single integer (see Section 11.3).

[3] This technique was not introduced in Part II of this thesis. Instead, we chose to base our presentation on the decisional Error-Free Approximate-GCD problem, first introduced in [KLYC13] and proven to be equivalent to the computational Error-Free Approximate-GCD problem in [CLT14a].

[4] In a recent work [LSS14], Langlois, Stehlé and Steinfield improved upon the GGH scheme [GGH13a] and proposed a third re-randomization technique.

11.2 Framework for Approximate Multilinear Maps

In this section, we first recall the formal definition of κ-linear maps as introduced in [BS03], and then describe the definition of *graded encoding schemes* of [GGH13a], that is the "approximate" multilinear maps (yet richer) notion of Garg, Gentry and Halevi. One of the main differences of this construction, compared to the generic multilinear maps from [BS03], is that encodings are *randomized* and only from the final evaluation can be extracted a *deterministic* function of the encoded values.

Finally, we recall the new hardness assumption introduced by [GGH13a]: the Graded Decisional Diffie-Hellman (GDDH) problem. This problem is a natural variant of the corresponding Diffie-Hellman problem from group-based cryptography.

11.2.1 Cryptographic Multilinear Maps

A cryptographic κ-linear map, as defined by Boneh and Silverberg in [BS03][5], is a map e from $\mathbb{G}_1 \times \cdots \times \mathbb{G}_\kappa \to \mathbb{G}_T$, where the $\mathbb{G}_i$ are cyclic (additively noted) groups of order p, such that: (1) for all $g_i \in \mathbb{G}_i$ with $i \leqslant \kappa$, for all $j \leqslant \kappa$, for all $a \in \mathbb{Z}_p$, we have $e(g_1, \ldots, a \cdot g_j, \ldots, g_\kappa) = a \cdot e(g_1, \ldots, g_\kappa)$, and (2) the map e is non degenerate, *i.e.* if the elements g_i's are generators of their respective groups, then $e(g_1, \ldots, g_\kappa)$ generates the target group $\mathbb{G}_T$.

Similarly to bilinear maps, in order to have useful applications, one needs that no efficient algorithm to compute discrete logarithms in any of the $\mathbb{G}_i$'s exists, and one usually needs the multilinear equivalent to the Decisional Diffie-Hellman problem to be hard.[6]

Definition 11.1 (Multilinear Decisional Diffie-Hellman)**.** For a symmetric κ-linear maps scheme (*i.e.* with $\mathbb{G}_1 = \cdots = \mathbb{G}_\kappa$) as described above, the Multilinear Decisional Diffie-Hellman problem is the problem to distinguish between the distributions

$$(\mathsf{params}, a_0 \cdot g, a_1 \cdot g, \ldots, a_\kappa \cdot g, \Big(\prod_{i=0}^{\kappa} a_i\Big) \cdot e(g, \ldots, g))$$

and

$$(\mathsf{params}, a_0 \cdot g, a_1 \cdot g, \ldots, a_\kappa \cdot g, a \cdot e(g, \ldots, g))$$

where $\mathsf{params} = (\mathbb{G}, p, e, g)$ with $\mathbb{G} = (g)$, and $a, a_0, a_1, \ldots, a_\kappa$ are uniformly random in $\mathbb{Z}_p$.

In other words, it should be hard from $\kappa + 1$ encodings to distinguish between an encoding of the product of the encoded values from a random encoding in $\mathbb{G}_T$.

The first candidate that approximates these multilinear maps scheme is due to Garg, Gentry and Halevi [GGH13a]. There are essentially two main differences with what precedes:

1. In bilinear pairings (and more generally cryptographic multilinear maps) we have a map $e\colon \mathbb{G}^\kappa \to \mathbb{G}_T$ that is linear with respect to all its κ inputs:

$$e(a_1 \cdot g, \ldots, a_\kappa \cdot g) = \Big(\prod_{i=1}^{\kappa} a_i\Big) \cdot e(g, \ldots, g)\ . \tag{11.2}$$

   One can view $a \cdot g$ as an "encoding" of the integer $a \in \mathbb{Z}_p$ over the group $\mathbb{G}$ of order p generated by g. The main difference in our setting is that encodings are now randomized. This means that an element $a \in R$ (where R is a ring that plays the role of the exponent space $\mathbb{Z}_p$) has many possible encodings; only the final multilinear map $e(a_1 \cdot g, \ldots, a_\kappa \cdot g)$ is a deterministic function of the a_i's only, and not on the randomness used to encode a_i into $a_i \cdot g$.

[5] Actually in [BS03], Boneh and Silverberg considered the symmetric case $\mathbb{G}_1 = \cdots = \mathbb{G}_\kappa$. The asymmetric case with different $\mathbb{G}_i$'s has later been considered, for example in [Rot13].

[6] Langlois, Stehlé and Steinfield [LSS14] used the search variant of the Graded Diffie-Hellman problem and the random oracle model to prove the security of the N-party key agreement and the attribute-based encryption scheme.

2. The second main difference is that to every encoding is now associated a *level*. At level 0 we have the "plaintext" ring elements $a \in R$, at level 1 we have the encoding $a \cdot g$, and by combining k such encodings $a_i \cdot g$ at level 1 one obtains a level-k encoding where the underlying elements a_i are homomorphically multiplied in R. The difference with "classical" cryptographic multilinear maps is that we can now multiply any (bounded) subset of encodings, instead of strictly κ at a time as with Equation (11.2). For encodings at level κ we have a special zero-testing parameter $\mathbf{p}_{zt}$ that can extract a deterministic function of the underlying ring elements. This enables us to define a degree-κ multilinear map for encodings at level 1.

In the rest of this section, we recall the formal definition of graded encoding schemes, and the approximate multilinear maps schemes of Garg, Gentry and Halevi [GGH13a]. For simplicity we only consider the symmetric case throughout the thesis; we refer to [GGH13a] for a more general framework that can handle the asymmetric case.

11.2.2 Graded Encoding System

Let us provide the formal definition of a κ-Graded Encoding System from [GGH13a].

Definition 11.2 (κ-Graded Encoding System [GGH13a])**.** A κ-Graded Encoding System for a ring R is a system of sets $\mathcal{S} = \{S_v^{(\alpha)} \in \{0,1\}^* : v \in \mathbb{N}, \alpha \in R\}$, with the following properties:

1. For every $v \in \mathbb{N}$, the sets $\{S_v^{(\alpha)} : \alpha \in R\}$ are disjoint.

2. There are binary operations $+$ and $-$ (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every $v \in \mathbb{N}$, and every $u_1 \in S_v^{(a_1)}$ and $u_2 \in S_v^{(a_2)}$, it holds that $u_1 + u_2 \in S_v^{(\alpha_1+\alpha_2)}$ and $u_1 - u_2 \in S_v^{(\alpha_1-\alpha_2)}$ where $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ are addition and subtraction in R.

3. There is an associative binary operation $\times$ (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every v_1, v_2 with $0 \leqslant v_1 + v_2 \leqslant \kappa$, and every $u_1 \in S_{v_1}^{(\alpha_1)}$ and $u_2 \in S_{v_2}^{(\alpha_2)}$, it holds that $u_1 \times u_2 \in S_{v_1+v_2}^{(\alpha_1 \cdot \alpha_2)}$ where $\alpha_1 \cdot \alpha_2$ is multiplication in R.

11.2.3 Multilinear Maps Procedures

In this section, we describe the procedures for manipulating encodings of the approximate multilinear maps scheme proposed by Garg, Gentry and Halevi in [GGH13a].

Instance Generation. The randomized $\mathsf{InstGen}(1^\lambda, 1^\kappa)$ takes as inputs the parameters λ and κ, and outputs $(\mathsf{params}, \mathbf{p}_{zt})$, where params is a description of a κ-Graded Encoding System as above, and $\mathbf{p}_{zt}$ is a zero-test parameter.

Ring Sampler. The randomized $\mathsf{samp}(\mathsf{params})$ outputs a "level-zero encoding" $a \in S_0^{(\alpha)}$ for a nearly uniform element $\alpha \in R$. Note that the encoding a does not need to be uniform in $S_0^{(\alpha)}$.

Encoding. The (possibly randomized) $\mathsf{enc}(\mathsf{params}, a)$ takes as input a level-zero encoding $a \in S_0^{(\alpha)}$ for some $\alpha \in R$, and outputs the level-one encoding $u \in S_1^{(\alpha)}$ for the same α.

Re-Randomization. The randomized $\mathsf{reRand}(\mathsf{params}, i, u)$ re-randomizes encodings relative to the same level i. Specifically, given an encoding $u \in S_v^{(\alpha)}$, it outputs another encoding $u' \in S_v^{(\alpha)}$. Moreover for any two $u_1,\ u_2 \in S_v^{(\alpha)}$, the output distributions of $\mathsf{reRand}(\mathsf{params}, i, u_1)$ and $\mathsf{reRand}(\mathsf{params}, i, u_2)$ are nearly the same.

Addition and negation. Given params and two encodings relative to the same level, $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, we have $\mathsf{add}(\mathsf{params}, u_1, u_2) \in S_i^{(\alpha_1+\alpha_2)}$ and $\mathsf{neg}(\mathsf{params}, u_1) \in S_i^{(-\alpha_1)}$. Below we write $u_1 + u_2$ and $-u_1$ as a shorthand for applying these procedures.

Multiplication. For $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_j^{(\alpha_2)}$, we have $\mathsf{mul}(\mathsf{params}, u_1, u_2) = u_1 \times u_2 \in S_{i+j}^{(\alpha_1 \cdot \alpha_2)}$.

Zero-test. The procedure $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u)$ outputs 1 if $u \in S_\kappa^{(0)}$ and 0 otherwise.

Extraction. The procedure extracts a random function of ring elements from their level-κ encoding. Namely $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u)$ outputs $s \in \{0,1\}^\lambda$, such that:

1. For any $\alpha \in R$ and $u_1,\ u_2 \in S_\kappa^{(\alpha)}$, $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_1) = \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_2)$.

2. The distribution $\{\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u) : \alpha \in R, u \in S_\kappa^{(\alpha)}\}$ is nearly uniform over $\{0,1\}^\lambda$.

This concludes the definition of the procedures. In [GGH13a] the authors consider a slightly relaxed definition of isZero and ext, where isZero can still output 1 even for some non-zero encoding u with negligible probability, and ext can extract different outputs when applied to encodings of the same elements, also with negligible probability; see [GGH13a] for the corresponding definitions.

11.2.4 Hardness Assumption

Finally we recall the new hardness assumptions for multilinear maps introduced in [GGH13a]. The Graded Decisional Diffie-Hellman is an analogue of the Multilinear Decisional Diffie-Hellman (Definition 11.1) for these approximate multilinear maps: given a set of $\kappa+1$ level-one encodings of random elements, it should be unfeasible to distinguish a level-κ encoding of their product from random.

Graded Decisional Diffie-Hellman (GDDH). Let $\mathcal{GE}$ be a graded encoding scheme consisting of all the routines above. For an adversary $\mathcal{A}$ and parameters λ, κ we consider the following process:

1. $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$

2. Choose $a_j \leftarrow \mathsf{samp}(\mathsf{params})$ for all $1 \leqslant j \leqslant \kappa+1$

3. Set $u_j \leftarrow \mathsf{reRand}(\mathsf{params}, 1, \mathsf{enc}(\mathsf{params}, 1, a_j))$ for all $1 \leqslant j \leqslant \kappa+1$ // encodings at level 1

4. Choose $b \leftarrow \mathsf{samp}(\mathsf{params})$ // encoding at level 0

5. Set $\tilde{u} = \mathsf{reRand}(\mathsf{params}, \kappa, \mathsf{enc}(\mathsf{params}, \kappa, \prod_{i=1}^{\kappa+1} a_i))$ // encoding of the right product at level κ

6. Set $\hat{u} = \mathsf{reRand}(\mathsf{params}, \kappa, \mathsf{enc}(\mathsf{params}, \kappa, b))$ // encoding of a random value at level κ

The GDDH distinguisher is given as input the $\kappa+1$ level-one encodings u_j and either $\tilde{u}$ (encoding the right product) or $\hat{u}$ (encoding a random value), and must decide which is the case. The GDDH assumption states that the advantage of any efficient adversary is negligible in the security parameter.

Remark 11.3. For the GDDH problem to be hard, it must hold that, for random and uniformly generated $a_i \in R$, $\prod_{i=1}^{\kappa+1} a_i \neq 0$ with overwhelming probability (otherwise one could distinguish by the procedure isZero).

Zero-Test Security. Garg, Gentry and Helvi also introduced a security notation related to the zero-testing parameter. This zero-test security notion states that either the isZero procedure outputs 1 with negligible probability when the encoding is not an encoding of 0 (statistical version), or that it should be hard to find such an encoding (computational version).

Definition 11.4 (Zero-Test Security)**.** Let $\mathcal{GE}$ be a graded encoding scheme consisting of all the routines above. We say that $\mathcal{GE}$ enjoys statistical zero-test security if, for parameters λ, κ, we have

$$\Pr[\exists u \notin S_\kappa^{(0)} : \mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u) = 1] \leqslant \mathsf{negl}(\lambda)\,.$$

We say that $\mathcal{GE}$ enjoys computational zero-test security if, for parameters λ, κ and any adversary $\mathcal{A}$, we have

$$\Pr_{\substack{(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa) \\ u \leftarrow \mathcal{A}(\mathsf{params}, \mathbf{p}_{zt})}} [\exists u \notin S_\kappa^{(0)} \text{ and } \mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u) = 1] \leqslant \mathsf{negl}(\lambda)\,.$$

Looking (far) ahead we note that our heuristic optimization in Chapter 12 that consists in reducing the number of elements in the zero-testing vector will yield a scheme that is not computationally zero-test secure – and therefore neither statistically zero-test secure –, as an adversary can use LLL to produce a level-κ encoding that will solve the security game.

11.3 Our new Encoding Scheme

System parameters. The main parameters are the security parameter λ and the required multilinearity level $\kappa \leqslant \mathsf{poly}(\lambda)$. Based on λ and κ, we choose the vector dimension n, the bit-size η of the primes p_i, the bit-size α of the primes g_i, the maximum bit-size ρ of the randomness used in encodings, and various other parameters that will be specified later; the constraints that these parameters must satisfy are described in the next section.

In our scheme a level-k encoding of a vector $\mathbf{m} = (m_i) \in \mathbb{Z}^n$ is an integer c such that for all $1 \leqslant i \leqslant n$:

$$c \equiv \frac{r_i \cdot g_i + m_i}{z^k} \pmod{p_i} \tag{11.3}$$

where the r_i's are ρ-bit random integers and the g_i's are α-bit primes (specific to the encoding c), with the following secret parameters: the p_i's are η-bit prime integers and the denominator z is a random (invertible) integer modulo $x_0 = \prod_{i=1}^n p_i$. The integer c is therefore defined by CRT modulo x_0, where x_0 is made public. Since the p_i's must remain secret, the user cannot encode the vectors $\mathbf{m} \in \mathbb{Z}^n$ by CRT directly from Equation (11.3); instead one includes in the public parameters a set of ℓ level-0 encodings x'_j of random vectors $\mathbf{a}_j \in \mathbb{Z}^n$, and the user can generate a random level-0 encoding by computing a random subset-sum of those x'_j's.

Remark 11.5. From Equation (11.3) each integer m_i is actually defined modulo g_i. Therefore our scheme encodes vectors $\mathbf{m}$ from the ring $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$.

Instance generation. $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$. We generate n secret random η-bit primes p_i and publish $x_0 = \prod_{i=1}^n p_i$. We generate a random invertible integer z modulo x_0. We generate n random α-bit prime integers g_i and a secret matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}^{n \times \ell}$, where each component a_{ij} is randomly generated in $[0, g_i) \cap \mathbb{Z}$. We generate an integer y, three sets of integers $\{x_j\}_{j=1}^\tau$, $\{x'_j\}_{j=1}^\ell$ and $\{\Pi_j\}_{j=1}^n$, a zero-testing vector $\mathbf{p}_{zt}$, and a seed s for a strong randomness extractor, as described later. We publish the parameters $\mathsf{params} = \big(n, \eta, \alpha, \rho, \beta, \tau, \ell, y, \{x_j\}_{j=1}^\tau, \{x'_j\}_{j=1}^\ell, \{\Pi_j\}_{j=1}^n, s\big)$ and $\mathbf{p}_{zt}$.

Sampling level-zero encodings. $c \leftarrow \mathsf{samp}(\mathsf{params})$. We publish as part as our instance generation a set of ℓ integers x'_j, where each x'_j encodes at level-0 the column vector $\mathbf{a}_j \in \mathbb{Z}^n$ of the secret matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}^{n \times \ell}$. More precisely, using the CRT modulo x_0 we generate integers x'_j such that:

$$1 \leqslant j \leqslant \ell, \qquad x'_j \equiv r'_{ij} \cdot g_i + a_{ij} \pmod{p_i} \tag{11.4}$$

where the r'_{ij}'s are randomly generated in $(-2^\rho, 2^\rho) \cap \mathbb{Z}$.

Our randomized sampling algorithm $\mathsf{samp}(\mathsf{params})$ works as follows: we generate a random binary vector $\mathbf{b} = (b_j) \in \{0,1\}^\ell$ and output the level-0 encoding

$$c = \sum_{j=1}^{\ell} b_j \cdot x'_j \bmod x_0 \ .$$

From Equation (11.4), this gives $c \equiv \big(\sum_{j=1}^\ell r'_{ij} b_j\big) \cdot g_i + \sum_{j=1}^\ell a_{ij} b_j \pmod{p_i}$; as required the output c is a level-0 encoding:

$$c \equiv r_i \cdot g_i + m_i \pmod{p_i} \tag{11.5}$$

of some vector $\mathbf{m} = \mathbf{A} \cdot \mathbf{b} \in \mathbb{Z}^n$ which is a random subset-sum of the column vectors $\mathbf{a}_j$. We note that for such level-0 encodings we get $|r_i \cdot g_i + m_i| \leqslant \ell \cdot 2^{\rho+\alpha}$ for all i.

The following Lemma states that, as required, the distribution of $\mathbf{m}$ can be made statistically close to uniform over $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$; the proof is based on applying the Leftover Hash Lemma over the set R.

Lemma 11.6. *Let $c \leftarrow \mathsf{samp}(\mathsf{params})$ and write $c \equiv r_i \cdot g_i + m_i \pmod{p_i}$. Assume $\ell \geqslant n \cdot \alpha + 2\lambda$. The distribution of $(\mathsf{params}, \mathbf{m})$ is statistically close to the distribution of $(\mathsf{params}, \mathbf{m}')$ where $\mathbf{m}' \leftarrow R$.*

Proof. Write $x'_j \equiv r'_{ij} \cdot g_i + a_{ij} \pmod{p_i}$ for $1 \leqslant j \leqslant \ell$ and $1 \leqslant i \leqslant n$. Each component a_{ij} is randomly generated in $[0, g_i) \cap \mathbb{Z}$; therefore the column vectors $\mathbf{a}_j$ of the matrix (a_{ij}) are randomly and independently generated in R. By the corollary of the Leftover Hash Lemma for abelian groups (Corollary 3.3), we have that $(\mathbf{a}_1, \ldots, \mathbf{a}_\ell, \mathbf{m})$ with $\mathbf{m} = \mathbf{A} \cdot \mathbf{b}$ is ε-uniform over $R^{\ell+1}$, where

$$\varepsilon = \frac{1}{2}\sqrt{\frac{|R|}{2^\ell}} \leqslant 2^{(\alpha \cdot n - \ell)/2} .$$

Therefore by taking $\ell \geqslant n \cdot \alpha + 2\lambda$ we obtain that the distribution of $(\mathsf{params}, \mathbf{m})$ is statistically close to the distribution of $(\mathsf{params}, \mathbf{m}')$ for $\mathbf{m}' \leftarrow R$. □

Encodings at higher levels. $c_k \leftarrow \mathsf{enc}(\mathsf{params}, k, c)$. To allow encoding at higher levels, we publish as part of our instance-generation a level-one random encoding of $\mathbf{1}$, namely an integer y such that:

$$y \equiv \frac{r_i \cdot g_i + 1}{z} \pmod{p_i}$$

for random integers $r_i \in (-2^\rho, 2^\rho) \cap \mathbb{Z}$; as previously the integer y is computed by CRT modulo x_0.

Given a level-0 encoding c of $\mathbf{m} \in \mathbb{Z}^n$ as given by Equation (11.5), we can then compute a level-1 encoding of the same $\mathbf{m}$ by computing $c_1 = c \cdot y \bmod x_0$. Namely we obtain as required:

$$c_1 \equiv \frac{r_i^{(1)} \cdot g_i + m_i}{z} \pmod{p_i} \tag{11.6}$$

for some integers $r_i^{(1)}$, and we get $|r_i^{(1)} \cdot g_i + m_i| \leqslant \ell \cdot 2^{2(\rho+\alpha)}$ for all i. More generally to generate a level-k encoding we compute $c_k = c_0 \cdot y^k \bmod x_0$.

In multipartite Diffie-Hellman key-exchange every user keeps a private level-0 encoding c and publishes a level-1 encoding of the same underlying (unknown) $\mathbf{m}$; however one cannot publish $c_1 = c \cdot y \bmod x_0$ directly since the private level-0 encoding c could be recovered immediately from $c = c_1/y \bmod x_0$. Instead the level-1 encoding c_1 must first be re-randomized into a new level-1 encoding c'_1 whose distribution does not depend on the original c as long as it encodes the same $\mathbf{m}$.

Re-randomization. $c' \leftarrow \mathsf{reRand}(\mathsf{params}, k, c)$. To allow re-randomization of encodings at level $k = 1$,[7] we publish as part of our instance-generation a set of n integers Π_j which are all level-1 random encodings of zero:

$$1 \leqslant j \leqslant n, \qquad \Pi_j \equiv \frac{\varpi_{ij} \cdot g_i}{z} \pmod{p_i} .$$

The matrix $\mathbf{\Pi} = (\varpi_{ij}) \in \mathbb{Z}^{n \times n}$ is a diagonally dominant matrix generated as follows: the non-diagonal entries are randomly and independently generated in $(-2^\rho, 2^\rho) \cap \mathbb{Z}$, while the diagonal entries are randomly generated in $(n2^\rho, n2^\rho + 2^\rho) \cap \mathbb{Z}$.[8]

We also publish as part of our instance-generation a set of τ integers x_j, where each x_j is a level-1 random encoding of zero:

$$1 \leqslant j \leqslant \tau, \qquad x_j \equiv \frac{r_{ij} \cdot g_i}{z} \pmod{p_i}$$

and where the column vectors of the matrix $(r_{ij}) \in \mathbb{Z}^{n \times \tau}$ are randomly and independently generated in the half-open parallelepiped spanned by the columns of the previous matrix $\mathbf{\Pi}$; see Appendix 11.A for a concrete algorithm.

Given as input a level-1 encoding c_1 as given by Equation (11.6), we randomize c_1 with a random subset-sum of the x_j's and a linear combination of the Π_j's:

$$c'_1 = c_1 + \sum_{j=1}^{\tau} b_j \cdot x_j + \sum_{j=1}^{n} b'_j \cdot \Pi_j \bmod x_0 \tag{11.7}$$

[7] One can easily adapt this procedure to randomize at level $k \geqslant 1$ by publishing additional similarly-defined integers.

[8] Note that we cannot take $\mathbf{\Pi} = (n2^\rho)\mathbf{I}_n$ because we publish encodings of the columns of $\mathbf{\Pi}$ and this would allow to factorize: we need to have random noises modulo each of the p_i's.

where $b_j \leftarrow \{0,1\}$, and $b'_j \leftarrow [0, 2^\mu) \cap \mathbb{Z}$. The following Lemma shows that as required the distribution of c'_1 is nearly independent of the input (as long as it encodes the same $\mathbf{m}$). This follows essentially from our "Leftover Hash Lemma over lattices"; see Section 3.3.1.

Lemma 11.7. *Let $c \leftarrow \mathsf{samp}(\mathsf{params})$, $c_1 \leftarrow \mathsf{enc}(\mathsf{params}, 1, c)$, and $c'_1 \leftarrow \mathsf{reRand}(\mathsf{params}, 1, c_1)$. Write $c'_1 \equiv (r_i \cdot g_i + m_i)/z \pmod{p_i}$ for all $1 \leqslant i \leqslant n$, and $\mathbf{r} = (r_1, \ldots, r_n)^t$. Let $\tau \geqslant n \cdot (\rho + \log_2(2n)) + 2\lambda$ and $\mu \geqslant \rho + \alpha + \lambda$. The distribution of $(\mathsf{params}, \mathbf{r})$ is statistically close to that of $(\mathsf{params}, \mathbf{r}')$, where $\mathbf{r}' \in \mathbb{Z}^n$ is randomly generated in the half-open parallelepiped spanned by the column vectors of $2^\mu \mathbf{\Pi}$.*

Writing $c'_1 \equiv (r'_i \cdot g_i + m_i)/z \pmod{p_i}$, and using $|r_{ij} \cdot g_i| \leqslant 2n2^{\rho+\alpha}$ for all i, j, we obtain $|r'_i \cdot g_i + m_i| \leqslant \ell 2^{2(\rho+\alpha)} + \tau \cdot 2n2^{\rho+\alpha} + n \cdot 2n2^{\mu+\rho+\alpha}$. Using $\mu \geqslant \rho + \alpha + \lambda$ this gives $|r'_i \cdot g_i + m_i| \leqslant 4n^2 2^{\mu+\rho+\alpha}$.

Adding and Multiplying Encodings. It is clear that one can homomorphically add encodings. Moreover the product of κ level-1 encodings u_i can be interpreted as an encoding of the product. Namely, given level-one encodings u_j of vectors $\mathbf{m}_j \in \mathbb{Z}^n$ for $1 \leqslant j \leqslant \kappa$, with $u_j \equiv (r_{ij} \cdot g_i + m_{ij})/z \pmod{p_i}$, we simply let:

$$u = \prod_{j=1}^{\kappa} u_j \bmod x_0 \ .$$

This gives:

$$u \equiv \frac{\prod_{j=1}^{\kappa} (r_{ij} \cdot g_i + m_{ij})}{z^\kappa} \equiv \frac{r_i \cdot g_i + \Big(\prod_{j=1}^{\kappa} m_{ij}\Big) \bmod g_i}{z^\kappa} \pmod{p_i}$$

for some $r_i \in \mathbb{Z}$. This is a level-κ encoding of the vector $\mathbf{m}$ obtained by componentwise product of the vectors $\mathbf{m}_j$, as long as $\prod_{j=1}^{\kappa}(r_{ij} \cdot g_i + m_{ij}) < p_i$ for all i. When computing the product of κ level-1 encodings from reRand and one level-0 encoding from samp (as in multipartite Diffie-Hellman key exchange, *cf.* Chapter 12), we obtain from previous bounds $|r_i| \leqslant (4n^2 2^{\mu+\rho+\alpha})^\kappa \cdot \ell \cdot 2^{\rho+1}$ for all i.

Zero Testing. $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u_\kappa) \stackrel{?}{=} 0/1$. We can test equality between encodings by subtracting them and testing for zero. To enable zero testing we randomly generate an integer matrix $\mathbf{H} = (h_{ij}) \in \mathbb{Z}^{n \times n}$ such that $\mathbf{H}$ is invertible in $\mathbb{Z}$ and both $\|\mathbf{H}^t\|_\infty \leqslant 2^\beta$ and $\|(\mathbf{H}^{-1})^t\|_\infty \leqslant 2^\beta$, for some parameter β specified later; here $\|\cdot\|_\infty$ is the operator norm on $n \times n$ matrices with respect to the ℓ^∞ norm on $\mathbb{R}^n$. A technique for generating such $\mathbf{H}$ is discussed in Appendix 11.B. We then publish as part of our instance generation the following zero-testing vector $\mathbf{p}_{zt} \in \mathbb{Z}^n$:

$$(\mathbf{p}_{zt})_j = \sum_{i=1}^{n} h_{ij} \cdot \big(z^\kappa \cdot g_i^{-1} \bmod p_i\big) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0 \ . \tag{11.8}$$

To determine whether a level-κ encoding c is an encoding of zero or not, we compute the vector $\boldsymbol{\omega} = c \cdot \mathbf{p}_{zt} \bmod x_0$ and test whether $\|\boldsymbol{\omega}\|_\infty$ is small:

$$\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, c) = \begin{cases} 1 & \text{if } \|c \cdot \mathbf{p}_{zt} \bmod x_0\|_\infty < x_0 \cdot 2^{-\nu} \\ 0 & \text{otherwise} \end{cases}$$

for some parameter ν specified later.

Namely for a level-κ ciphertext c we have $c \equiv (r_i \cdot g_i + m_i)/z^\kappa \pmod{p_i}$ for some $r_i \in \mathbb{Z}$; therefore for all $1 \leqslant i \leqslant n$ we can write:

$$c = q_i \cdot p_i + (r_i \cdot g_i + m_i) \cdot \big(z^{-\kappa} \bmod p_i\big) \tag{11.9}$$

for some $q_i \in \mathbb{Z}$. Therefore combining Equations (11.8) and (11.9), we get:

$$(\boldsymbol{\omega})_j = (c \cdot \mathbf{p}_{zt} \bmod x_0)_j = \sum_{i=1}^{n} h_{ij} \cdot \big(r_i + m_i \cdot (g_i^{-1} \bmod p_i)\big) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0 \ . \tag{11.10}$$

Therefore if $m_i = 0$ for all $1 \leqslant i \leqslant n$, then $\|\boldsymbol{\omega}\|_\infty = \|c \cdot \mathbf{p}_{zt} \bmod x_0\|_\infty$ is small compared to x_0 when the r_i's are small enough, *i.e.* a limited number of additions/multiplications on encodings has been performed. Conversely if $m_i \neq 0$ for some i we show that $\|\boldsymbol{\omega}\|_\infty$ must be large. More precisely we have the following lemma.

Lemma 11.8. *Let n, η, α and β be as in our parameter setting. Let ρ_f be such that $\rho_f+\lambda+\alpha+2\beta \leqslant \eta-8$, and let $\nu = \eta-\beta-\rho_f-\lambda-3 \geqslant \alpha+\beta+5$. Let c be such that $c \equiv (r_i \cdot g_i + m_i)/z^\kappa \pmod{p_i}$ for all $1 \leqslant i \leqslant n$, where $0 \leqslant m_i < g_i$ for all i. Let $\mathbf{r} = (r_i)_{1\leqslant i\leqslant n}$ and assume that $\|\mathbf{r}\|_\infty < 2^{\rho_f}$. If $\mathbf{m} = 0$ then $\|\boldsymbol{\omega}\|_\infty < x_0 \cdot 2^{-\nu-\lambda-2}$. Conversely if $\mathbf{m} \neq 0$ then $\|\boldsymbol{\omega}\|_\infty \geqslant x_0 \cdot 2^{-\nu+2}$.*

Proof. We have assumed

$$\rho_f + \lambda + \alpha + 2\beta \leqslant \eta - 8 \,,$$

which gives:

$$\nu = \eta - \beta - \rho_f - \lambda - 3 \geqslant \alpha + \beta + 5 \,. \tag{11.11}$$

We consider the vector $\mathbf{R} = (R_i)_{1\leqslant i\leqslant n}$ where:

$$R_i = ((r_i + m_i \cdot g_i^{-1}) \bmod p_i) \cdot (x_0/p_i) \,. \tag{11.12}$$

Equation (11.10) can then be written:

$$\boldsymbol{\omega} = \mathbf{H}^t \cdot \mathbf{R} \bmod x_0 \,. \tag{11.13}$$

If $\mathbf{m} = \mathbf{0}$ then we have $R_i = r_i \cdot x_0/p_i$ for all i, which gives using $p_i \geqslant 2^{\eta-1}$ for all i:

$$\|\mathbf{R}\|_\infty \leqslant \|\mathbf{r}\|_\infty \cdot \max_{1\leqslant i\leqslant n}(x_0/p_i) \leqslant \|\mathbf{r}\|_\infty \cdot x_0 \cdot 2^{-\eta+1} \,.$$

Since by definition $-p/2 < (z \bmod p) \leqslant p/2$, we have $|z \bmod p| \leqslant |z|$ for any z, p; therefore we obtain from Equation (11.13) using $\|\mathbf{r}\|_\infty < 2^{\rho_f}$

$$\|\boldsymbol{\omega}\|_\infty = \|\mathbf{H}^t \cdot \mathbf{R} \bmod x_0\|_\infty \leqslant \|\mathbf{H}^t \cdot \mathbf{R}\|_\infty \leqslant \|\mathbf{H}^t\|_\infty \cdot \|\mathbf{R}\|_\infty < x_0 \cdot 2^{\beta+\rho_f-\eta+1} = x_0 \cdot 2^{-\nu-\lambda-2} \,.$$

Conversely assume that $\|\boldsymbol{\omega}\|_\infty < x_0 \cdot 2^{-\nu+2}$. From Equation (11.13) we have:

$$\mathbf{R} \equiv (\mathbf{H}^{-1})^t \cdot \boldsymbol{\omega} \pmod{x_0} \,. \tag{11.14}$$

From Equation (11.12) we have $\|\mathbf{R}\|_\infty < x_0/2$. Moreover from Equation (11.11) we have $\nu - \beta \geqslant \alpha + 5$, which gives

$$\|(\mathbf{H}^{-1})^t \cdot \boldsymbol{\omega}\|_\infty \leqslant \|(\mathbf{H}^{-1})^t\|_\infty \cdot \|\boldsymbol{\omega}\|_\infty \leqslant x_0 \cdot 2^{\beta-\nu+2} \leqslant x_0 \cdot 2^{-\alpha-3} < x_0/2 \,. \tag{11.15}$$

This shows that Equation (11.14) holds in $\mathbb{Z}$ and not only modulo x_0; therefore we must have

$$\|\mathbf{R}\|_\infty \leqslant x_0 \cdot 2^{\beta-\nu+2} \,.$$

Letting $v_i = (r_i + m_i \cdot g_i^{-1}) \bmod p_i$ for $1 \leqslant i \leqslant n$, this gives $|v_i| \cdot (x_0/p_i) \leqslant x_0 \cdot 2^{\beta-\nu+2}$, and therefore $|v_i| \leqslant p_i \cdot 2^{\beta-\nu+2}$ for all i. We have $g_i \cdot (v_i - r_i) \equiv m_i \pmod{p_i}$; we show that the equality actually holds over $\mathbb{Z}$. Namely for all i we have $|m_i| < g_i < p_i/2$ and with $g_i < 2^\alpha$ we get:

$$|g_i \cdot (v_i - r_i)| \leqslant g_i \cdot (|v_i| + |r_i|) \leqslant p_i \cdot 2^{\alpha+\beta-\nu+2} + 2^{\alpha+\rho_f} \leqslant p_i/8 + p_i/8 < p_i/2$$

which implies that the equality $g_i \cdot (v_i - r_i) = m_i$ holds over $\mathbb{Z}$. Therefore $m_i \equiv 0 \pmod{g_i}$ for all i, which implies $\mathbf{m} = \mathbf{0}$. □

Extraction. $sk \leftarrow \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_\kappa)$. This part is essentially the same as in [GGH13a]. To extract a random function of the vector $\mathbf{m}$ encoded in a level-κ encoding c, we multiply c by the zero-testing parameter $\mathbf{p}_{zt}$ modulo x_0, collect the ν most significant bits of each of the n components of the resulting vector, and apply a strong randomness extractor (using the seed s from params):

$$\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, c) = \mathsf{Extract}_s\big(\mathsf{msbs}_\nu(c \cdot \mathbf{p}_{zt} \bmod x_0)\big)$$

where msbs_ν extracts the ν most significant bits of the result. Namely if two encodings c and c' encode the same $\mathbf{m} \in \mathbb{Z}^n$ then from Lemma 11.8 we have $\|(c - c') \cdot \mathbf{p}_{zt} \bmod x_0\|_\infty < x_0 \cdot 2^{-\nu-\lambda-2}$, and therefore we expect that $\boldsymbol{\omega} = c \cdot \mathbf{p}_{zt} \bmod x_0$ and $\boldsymbol{\omega'} = c' \cdot \mathbf{p}_{zt} \bmod x_0$ agree on their ν most significant bits, and therefore extract to the same value.[9]

Conversely if c and c' encode different vectors then by Lemma 11.8 we must have $\|(c - c') \cdot \mathbf{p}_{zt} \bmod x_0\|_\infty > x_0 \cdot 2^{-\nu+2}$, and therefore the ν most significant bits of the corresponding $\boldsymbol{\omega}$ and $\boldsymbol{\omega'}$ must be different. This implies that for random $\mathbf{m} \in R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ the min-entropy of $\mathsf{msbs}_\nu(c \cdot \mathbf{p}_{zt} \bmod x_0)$ when c encodes $\mathbf{m}$ is at least $\log_2 |R| \geqslant n(\alpha - 1)$. Therefore we can use a strong randomness extractor to extract a nearly uniform bit-string of length $\lfloor \log_2 |R| \rfloor - \lambda$.

This concludes the description of our multilinear encoding scheme. In Section 11.3.3 we provide a comparison with the original scheme from [GGH13a].

11.3.1 Setting the Parameters

The constraints on our system are really similar to the constraints on the multi-slot DGHV scheme described in Chapter 7. In particular, the system parameters must satisfy the following constraints:

- The bit-size ρ of the randomness used for encodings must satisfy $\rho = \Omega(\lambda)$ to avoid brute force attack on the noise, including the improved attack from [CN12] with complexity $\tilde{\mathcal{O}}(2^{\rho/2})$.

- The bit-size α of the primes g_i must be large enough so that the order of the group $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ does not contain small prime factors; this is required for the GDDH problem to be hard (*cf.* Remark 11.3). One can take $\alpha = \lambda$.

- The parameter n must be large enough to thwart lattice-based attacks on the encodings, namely $n = \omega(\eta \log \lambda)$; see Section 11.5.1.

- The number ℓ of level-0 encodings x'_j for samp must satisfy $\ell \geqslant n \cdot \alpha + 2\lambda$ in order to apply the leftover hash lemma; see Lemma 11.6.

- The number τ of level-1 encodings x_j must satisfy $\tau \geqslant n \cdot (\rho + \log_2(2n)) + 2\lambda$ in order to apply our "leftover hash lemma over lattices". For the same reason the bit-size μ of the linear sum coefficients must satisfy $\mu \geqslant \alpha + \rho + \lambda$; see Lemma 11.7.

- The bitsize β of the matrix $\mathbf{H}$ entries must satisfy $\beta = \Omega(\lambda)$ in order to avoid the GCD attack from Section 11.5.2. One can take $\beta = \lambda$.

- The bit-size η of the primes p_i must satisfy $\eta \geqslant \rho_f + \alpha + 2\beta + \lambda + 8$, where ρ_f is the maximum bit size of the randoms r_i a level-κ encoding (see Lemma 11.8). When computing the product of κ level-1 encodings and an additional level-0 encoding (as in a multipartite Diffie-Hellman key exchange with $N = \kappa + 1$ users), one obtains $\rho_f = \kappa \cdot (\mu + \rho + \alpha + 2\log_2 n + 2) + \rho + \log_2 \ell + 1$ (see previous Section).

- The number ν of most significant bits to extract can then be set to $\nu = \eta - \beta - \rho_f - \lambda - 3$ (see Lemma 11.8).

[9] Two coefficients ω and ω' from $\boldsymbol{\omega}$ and $\boldsymbol{\omega'}$ could still be on the opposite sides of a boundary, with $\lfloor \omega/2^k \rfloor = v$ and $\lfloor \omega'/2^k \rfloor = v + 1$, so that ω and ω' would extract to different MSBs v and $v + 1$. Heuristically this happens with probability $\mathcal{O}(2^{-\lambda})$. The argument can be made rigorous by generating a public random integer W modulo x_0 in the parameters, and extracting the MSBs of $\omega + W \bmod x_0$ instead of $\omega \bmod x_0$ for all coefficients ω of the vector $\boldsymbol{\omega}$.

11.3.2 Security of our Construction

As in [GGH13a] the security of our construction relies on new assumptions that do not seem to be reducible to more classical assumptions. Namely, as in [GGH13a] one can make the assumption that solving the Graded Decisional Diffie-Hellman problem (GDDH) recalled in Section 11.2.4 is hard in our scheme. This enables to prove the security of the one-round N-way Diffie-Hellman key exchange protocol (*cf.* Chapter 12). Ideally one would like to reduce such assumption to a more classical assumption, such as the Approximate-GCD assumption, but that does not seem feasible. Therefore to gain more confidence in our scheme we describe various attacks in Section 11.5.

11.3.3 Comparison with GGH Multilinear Maps

In this section, we rewrite our scheme using exactly the same notations as in [GGH13a] whenever possible, to better highlight the similarities.

The construction in [GGH13a] works in the polynomial ring $R = \mathbb{Z}[X]/(X^n + 1)$, where n is large enough to ensure security. One generates a secret short ring element $\mathbf{g} \in R$, generating a principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle \subset R$. One also generates an integer parameter q and another random secret $\mathbf{z} \in R/qR$. One encodes elements of the quotient ring $R/\mathcal{I}$, namely elements of the form $\mathbf{e} + \mathcal{I}$ for some $\mathbf{e}$, as follows: a level-i encoding of the coset $\mathbf{e} + \mathcal{I}$ is an element of the form $u_k = [\mathbf{c}/\mathbf{z}^i]_q$, where $\mathbf{c} \in \mathbf{e} + \mathcal{I}$ is short. Such encodings can be both added and multiplied, as long as the norm of the numerators remain shorter than q; in particular the product of κ encodings at level 1 gives an encoding at level κ. For such level-κ encodings one can then define a zero-testing parameter $\mathbf{p}_{zt} = [\mathbf{h}\mathbf{z}^\kappa/\mathbf{g}]_q$, for some small $\mathbf{h} \in R$. Then given a level-κ encoding $\mathbf{u} = [\mathbf{c}/\mathbf{z}^\kappa]$ one can compute $[\mathbf{p}_{zt} \cdot \mathbf{u}]_q = [\mathbf{h}\mathbf{c}/\mathbf{g}]_q$; when $\mathbf{c}$ is an encoding of zero we have $\mathbf{c}/\mathbf{g} \in R$, which implies that $\mathbf{h}\mathbf{c}/\mathbf{g}$ is small in R, and therefore $[\mathbf{h}\mathbf{c}/\mathbf{g}]_q$ is small; this provides a way to test whether a level-κ encoding $\mathbf{c}$ is an encoding of 0.

In our construction one could write $R = \mathbb{Z}^n$, and define a secret short ring element $\mathbf{g} \in R$, generating a principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle \subset R$, which gives $\mathcal{I} = (g_i\mathbb{Z})_{1 \leqslant i \leqslant n}$. We also generate a ring element $\mathbf{p} \in R$ and let the principal ideal $\mathcal{J} = \langle \mathbf{p} \rangle \subset R$, which gives $\mathcal{J} = (p_i\mathbb{Z})_{1 \leqslant i \leqslant n}$. We let $q := x_0 = \prod_{i=1}^n p_i$ and for convenience we denote by $[\mathbf{u}]_q$ the CRT isomorphism from $R/\mathcal{J}$ to $\mathbb{Z}_q$. As in [GGH13a], in our scheme a level-i encoding of the coset $\mathbf{e}_\mathcal{I} = \mathbf{e} + \mathcal{I}$ is an element of the form $u = [\mathbf{c}/\mathbf{z}^i]_q$ where $\mathbf{c} \in \mathbf{e}_\mathcal{I}$ is short. Such encodings can be both added and multiplied, by working over the integers via the CRT isomorphism $[\cdot]_q$.

However, we cannot apply the zero-testing procedure from [GGH13a] in a straightforward way. Namely one could define the zero-testing parameter $p_{zt} = [\mathbf{h}\mathbf{z}^\kappa/\mathbf{g}]_q$ as in [GGH13a] where $\mathbf{h} \in \mathbb{Z}^n$ is a relatively small ring element. As in [GGH13a] given a level-κ encoding $u = [\mathbf{c}/\mathbf{z}^\kappa]_q$ one would compute the element:

$$\omega = p_{zt} \cdot u = \left[\frac{\mathbf{h}\mathbf{z}^\kappa}{\mathbf{g}} \cdot \frac{\mathbf{c}}{\mathbf{z}^\kappa} \right]_q = \left[\frac{\mathbf{h}\mathbf{c}}{\mathbf{g}} \right]_q . \tag{11.16}$$

As in [GGH13a] if u is an encoding of 0 then $\mathbf{c}$ is a multiple of $\mathbf{g}$ over $\mathbb{Z}^n$ hence $\mathbf{c}/\mathbf{g} \in \mathbb{Z}^n$ is short and therefore the vector $\mathbf{h}\mathbf{c}/\mathbf{g} \in \mathbb{Z}^n$ is short. However this does not imply that the corresponding integer ω obtained by CRT in Equation (11.16) is small, and we do not have a simple way of identifying integers whose reductions modulo the unknown p_i's are small.

Instead, we can define a slightly different notation: we consider the following additive homomorphism $R/\mathcal{J} \longrightarrow \mathbb{Z}_q$

$$\mathbf{u} \to \{\mathbf{u}\}_q = \sum_{i=1}^n u_i \cdot \prod_{i' \neq i} p_{i'} \bmod q$$

where as before $q = x_0 = \prod_{i=1}^n p_i$ and we define the zero-testing parameter:

$$p_{zt} := \{\mathbf{h}\mathbf{z}^\kappa/\mathbf{g}\}_q$$

As in [GGH13a] given a level-κ encoding $u = [\mathbf{c}/\mathbf{z}^\kappa]_q$ one can compute the element:

$$\omega = p_{zt} \cdot u \bmod q = \left\{ \frac{\mathbf{h}\mathbf{z}^\kappa}{\mathbf{g}} \right\}_q \cdot \left[\frac{\mathbf{c}}{\mathbf{z}^\kappa} \right]_q = \left\{ \frac{\mathbf{h}\mathbf{z}^\kappa}{\mathbf{g}} \cdot \frac{\mathbf{c}}{\mathbf{z}^\kappa} \right\}_q = \left\{ \frac{\mathbf{h}\mathbf{c}}{\mathbf{g}} \right\}_q$$

As in [GGH13a] if u is an encoding of 0 then $\mathbf{c}$ is a multiple of $\mathbf{g}$ over $\mathbb{Z}^n$ hence $\mathbf{c}/\mathbf{g} \in \mathbb{Z}^n$ is short and therefore the vector $\mathbf{hc}/\mathbf{g} \in \mathbb{Z}^n$ is short; this time, this implies that $\omega = \{\mathbf{hc}/\mathbf{g}\}_q$ is a short integer.

11.4 Another Leftover Hash Lemma over Lattices

As mentioned in the introduction, to prove the full randomization of encodings (Lemma 11.7) one cannot apply the classical Leftover Hash Lemma (*cf.* Section 3.3.1) because the noise in the encodings lives in some infinite ring instead of a finite group. In [GGH13a] the issue was solved by using linear sums with Gaussian coefficients. Namely the analysis in [AGHS13] shows that the resulting sum has a Gaussian distribution (over some lattice). As noted by the authors this technique can be seen as a "Leftover Hash Lemma over lattices". Such a technique would be applicable to our scheme as well.

In this section we describe an alternative technique (without Gaussian coefficients) that can also be seen as a "Leftover Hash Lemma over lattices". It consists in working modulo a lattice $L \subset \mathbb{Z}^n$ and applying the classical Leftover Hash Lemma over the finite group $\mathbb{Z}^n/L$. In this section, we provide a more formal description; namely we clearly state our "Leftover Hash Lemma over lattices" so that it can later be applied as a black-box (as the corresponding Theorem 3 in [AGHS13]). Namely our quotient lattice technique can independently be applied to the original encoding scheme from [GGH13a].

11.4.1 Leftover Hash Lemma over Lattices

Let $L \subset \mathbb{Z}^n$ be a lattice of rank n of basis $\mathcal{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$. Then every $\mathbf{x} \subset \mathbb{Z}^n$ can be uniquely written as:

$$\mathbf{x} = \xi_1 \mathbf{b}_1 + \ldots + \xi_n \mathbf{b}_n$$

for some real numbers ξ_i. Moreover, for every vector $\mathbf{x} \in \mathbb{Z}^n$ there is a unique $\mathbf{a} \in L$ such that:

$$\mathbf{y} = \mathbf{x} - \mathbf{a} = \xi'_1 \mathbf{b}_1 + \ldots + \xi'_n \mathbf{b}_n$$

where $0 \leqslant \xi'_i < 1$; we write $\mathbf{y} = \mathbf{x} \bmod \mathcal{B}$. Therefore each vector of $\mathbb{Z}^n/L$ has a unique representative in the half-open parallelepiped defined by the previous equation.

We denote by $D_{\mathcal{B}}$ the distribution obtained by generating a random element in the quotient $\mathbb{Z}^n/L$ and taking its unique representative in the half-open parallelepiped generated by the basis $\mathcal{B}$. Given a basis $\mathcal{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ and $\mu \in \mathbb{Z}^*$ we denote by $\mu\mathcal{B}$ the basis $(\mu\mathbf{b}_1, \ldots, \mu\mathbf{b}_n)$. For simplicity of notation, if $\mathbf{B}$ is the matrix whose columns are the $\mathbf{b}_i$'s, we also denote $D_{\mathbf{B}}$ the distribution $D_{\mathcal{B}}$. We are now ready to state our "Leftover Hash Lemma over Lattices".

Lemma 11.9. *Let $L \subset \mathbb{Z}^n$ be a lattice of rank n of basis $\mathcal{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$. Let $\mathbf{x}_i$ for $1 \leqslant i \leqslant m$ be generated independently according to the distribution $D_{\mathcal{B}}$. Set $s_1, \ldots, s_m \leftarrow \{0, 1\}$ and $t_1, \ldots, t_n \leftarrow \mathbb{Z} \cap [0, 2^\mu)$. Let $\mathbf{y} = \sum_{i=1}^m s_i \mathbf{x}_i + \sum_{i=1}^n t_i \mathbf{b}_i$ and $\mathbf{y}' \leftarrow D_{2^\mu \mathcal{B}}$. Then the distributions $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{y})$ and $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{y}')$ are ε-statistically close, with $\varepsilon = mn \cdot 2^{-\mu} + 1/2\sqrt{|\det L|/2^m}$.*

Proof. We consider the intermediate variable:

$$\mathbf{y}'' = \left(\sum_{i=1}^m s_i \mathbf{x}_i \bmod \mathcal{B} \right) + \sum_{i=1}^n t_i \mathbf{b}_i \,. \tag{11.17}$$

Firstly by applying the Leftover Hash Lemma over the finite abelian group $G = \mathbb{Z}^n/L$ (*cf.* Corollary 3.3), we obtain that the distributions $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \sum_{i=1}^m s_i \mathbf{x}_i \bmod \mathcal{B})$ and $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \boldsymbol{\psi})$ are ε_1-statistically close, where $\boldsymbol{\psi} \leftarrow D_{\mathcal{B}}$ and

$$\varepsilon_1 = 1/2\sqrt{|G|/2^m} = 1/2\sqrt{|\det(L)|/2^m} \,.$$

This implies that the distributions $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{y}'')$ and $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{y}')$ are also ε_1-statistically close.

Secondly we write:

$$\sum_{i=1}^{m} s_i \mathbf{x}_i \bmod \mathcal{B} = \sum_{i=1}^{m} s_i \mathbf{x}_i - \sum_{j=1}^{n} \chi_j \mathbf{b}_j \tag{11.18}$$

where $\chi_j \in \mathbb{Z}$ for all j. We also write $\mathbf{x_i} = \sum_j \xi_{ij} \mathbf{b}_j$ where by definition $0 \leqslant \xi_{ij} < 1$ for all i, j. This gives:

$$\sum_{i=1}^{m} s_i \mathbf{x}_i \bmod \mathcal{B} = \sum_{i=1}^{m} s_i \sum_{j=1}^{n} \xi_{ij} \mathbf{b}_j - \sum_{j=1}^{n} \chi_j \mathbf{b}_j = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} s_i \xi_{ij} - \chi_j \right) \mathbf{b}_j \,,$$

which implies $0 \leqslant \sum_{i=1}^{m} s_i \xi_{ij} - \chi_j < 1$ for all j, and therefore $0 \leqslant \chi_j \leqslant m$ for all j. Combining Equations (11.17) and (11.18) we have:

$$\mathbf{y}'' = \sum_{i=1}^{m} s_i \mathbf{x}_i + \sum_{i=1}^{n} (t_i - \chi_i) \mathbf{b}_i \,,$$

where as shown above $0 \leqslant \chi_i \leqslant m$ for all i. This implies that the distributions $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{y})$ and $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{y}'')$ are ε_2-statistically close, with $\varepsilon_2 = mn2^{-\mu}$. Therefore the distributions $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{y})$ and $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{y}')$ are $(\varepsilon_1 + \varepsilon_2)$-statistically close, which proves the lemma. □

We also show that the previous distribution $D_{2^\mu \mathcal{B}}$ is not significantly modified when a small vector $\mathbf{z} \in \mathbb{Z}^n$ is added and the operator norm of the corresponding matrix $\mathbf{B}^{-1}$ is upper-bounded.

Lemma 11.10. *Let $L \subset \mathbb{Z}^n$ be a full-rank lattice of basis $\mathcal{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, and let $\mathbf{B} \in \mathbb{Z}^{n \times n}$ be the matrix whose column vectors are the $\mathbf{b}_i$'s. For any $\mathbf{z} \in \mathbb{Z}^n$, the distribution of $\mathbf{z} + D_{2^\mu \mathcal{B}}$ is ε-statistically close to that of $D_{2^\mu \mathcal{B}}$, where $\varepsilon = 2^{-\mu} \cdot (\|\mathbf{z}\|_\infty \cdot \|\mathbf{B}^{-1}\|_\infty + 1)$.*

Proof. Let $\mathbf{u} \leftarrow D_{2^\mu \mathcal{B}}$ and $\mathbf{u}'' \leftarrow \mathbf{z} + D_{2^\mu \mathcal{B}}$. We can write:

$$\begin{aligned} \mathbf{u} &= \mathbf{v} + \sum_{i=1}^{n} s_i \mathbf{b}_i \\ \mathbf{u}'' &= \mathbf{z} + \mathbf{v} + \sum_{i=1}^{n} s_i \mathbf{b}_i \end{aligned}$$

where $\mathbf{v} \leftarrow D_{\mathcal{B}}$ and $s_i \leftarrow [0, 2^\mu) \cap \mathbb{Z}$. We consider the intermediate variable:

$$\mathbf{u}' = ((\mathbf{z} + \mathbf{v}) \bmod \mathcal{B}) + \sum_{i=1}^{n} s_i \mathbf{b}_i \,.$$

The distribution of $\mathbf{u}$ and $\mathbf{u}'$ are clearly the same. Let $\boldsymbol{\psi} = \mathbf{z} + \mathbf{v}$. We have:

$$\boldsymbol{\psi} \bmod \mathcal{B} = \boldsymbol{\psi} - \mathbf{B} \cdot \lfloor \mathbf{B}^{-1} \cdot \boldsymbol{\psi} \rfloor = \boldsymbol{\psi} - \sum_{i=1}^{n} t_i \mathbf{b}_i$$

where $\mathbf{t} = \lfloor \mathbf{B}^{-1} \cdot \boldsymbol{\psi} \rfloor$. This gives:

$$\mathbf{u}' = \mathbf{z} + \mathbf{v} + \sum_{i=1}^{n} (s_i - t_i) \mathbf{b}_i \,.$$

We have $\mathbf{t} = \lfloor \mathbf{B}^{-1} \cdot \mathbf{z} + \mathbf{B}^{-1} \cdot \mathbf{v} \rfloor$. Since $\mathbf{v}$ is in the half-open parallelepiped spanned by $\mathcal{B}$ we have that the components of $\mathbf{B}^{-1} \cdot \mathbf{v}$ are in $[0, 1)$, which gives:

$$\|\mathbf{t}\|_\infty \leqslant \|\mathbf{B}^{-1} \cdot \mathbf{z}\|_\infty + 1 \leqslant \|\mathbf{B}^{-1}\|_\infty \cdot \|\mathbf{z}\|_\infty + 1 \,.$$

Therefore the variables $\mathbf{u}'$ and $\mathbf{u}''$ are ε-statistically close, with $\varepsilon = 2^{-\mu} \left(\|\mathbf{B}^{-1}\|_\infty \cdot \|\mathbf{z}\|_\infty + 1 \right)$. This proves the Lemma. □

11.4.2 Re-Randomization of Encodings: Proof of Lemma 11.7

We are now ready to apply our "Leftover Hash Lemma over lattices" to prove the full randomization of encodings as stated in Lemma 11.7. Namely the re-randomization Equation (11.7) can be rewritten in vector form as:

$$\mathbf{r}' = \mathbf{r} + \mathbf{X} \cdot \mathbf{b} + \mathbf{\Pi} \cdot \mathbf{b}'$$

where $\mathbf{b} \leftarrow \{0,1\}^\tau$ and $\mathbf{b}' \leftarrow ([0, 2^\mu) \cap \mathbb{Z})^n$, and the columns of the matrix $\mathbf{X} \in \mathbb{Z}^{n \times \tau}$ are uniformly and independently generated in the parallelepiped spanned by the columns of the matrix $\mathbf{\Pi} \in \mathbb{Z}^{n \times n}$. To conclude, it therefore suffices to apply Lemma 11.9 and Lemma 11.10, using additionally an upper bound on $\|\mathbf{\Pi}^{-1}\|_\infty$. For this we use the fact that $\mathbf{\Pi}$ has been generated as a diagonally dominant matrix.

Given a matrix $\mathbf{B} = (b_{ij}) \in \mathbb{R}^{n \times n}$, we let $\Lambda_i(\mathbf{B}) = \sum_{k \neq i} |b_{ik}|$. A matrix $\mathbf{B} = (b_{ij}) \in \mathbb{R}^{n \times n}$ is said to be *diagonally dominant* if $|b_{ii}| > \Lambda_i(\mathbf{B})$ for all i. We recall the following facts for diagonally dominant matrices [Var75, Pri51].

Lemma 11.11. *Let $\mathbf{B} = (b_{ij}) \in \mathbb{R}^{n \times n}$ be a diagonally dominant matrix. Then the matrix $\mathbf{B}$ is invertible and $\|\mathbf{B}^{-1}\|_\infty \leqslant \max_{i=1,\ldots,n} \left(|b_{ii}| - \Lambda_i(\mathbf{B})\right)^{-1}$ where $\|\cdot\|_\infty$ is the operator norm on $n \times n$ matrices with respect to the ℓ^∞ norm on $\mathbb{R}^n$.*

Lemma 11.12. *Let $\mathbf{B} = (b_{ij}) \in \mathbb{R}^{n \times n}$ be a diagonally dominant matrix. Then*

$$\prod_{i=1}^{n} \left(|b_{ii}| - \Lambda_i(\mathbf{B})\right) \leqslant |\det \mathbf{B}| \leqslant \prod_{i=1}^{n} \left(|b_{ii}| + \Lambda_i(\mathbf{B})\right) .$$

Proof of Lemma 11.7. We write $c_1 \equiv \left(r_i^{(1)} \cdot g_i + m_i\right)/z \bmod p_i$ for all $1 \leqslant i \leqslant n$ and define $\mathbf{r^{(1)}} = (r_i^{(1)}) \in \mathbb{Z}^n$. We also write $x_j \equiv r_{ij} \cdot g_i/z \pmod{p_i}$ and $\mathit{\Pi}_j \equiv \varpi_{ij} \cdot g_i/z \pmod{p_i}$ and define the matrix $\mathbf{X} = (r_{ij}) \in \mathbb{Z}^{n \times \tau}$ and $\mathbf{\Pi} = (\varpi_{ij}) \in \mathbb{Z}^{n \times n}$. From the re-randomization equation (11.7), we can write:

$$\mathbf{r} = \mathbf{r^{(1)}} + \mathbf{X} \cdot \mathbf{b} + \mathbf{\Pi} \cdot \mathbf{b}' ,$$

where $\mathbf{b} \leftarrow \{0,1\}^\tau$, and $\mathbf{b}' \leftarrow ([0, 2^\mu) \cap \mathbb{Z})^n$.

Since at instance generation the columns of $\mathbf{X}$ are generated uniformly and independently in the parallelepiped spanned by the columns of $\mathbf{\Pi}$, applying our "Leftover Hash Lemma over lattices" (Lemma 11.9) we obtain that the distribution of $(\mathsf{params}, \mathbf{r})$ is ε_1-close to the distribution of $(\mathsf{params}, \mathbf{r^{(1)}} + D_{2^\mu \mathbf{\Pi}})$, with

$$\varepsilon_1 = \tau n 2^{-\mu} + \frac{1}{2}\sqrt{|\det \mathbf{\Pi}|/2^\tau} .$$

Since $\mathbf{\Pi}$ is a diagonally dominant matrix, we obtain from Lemma 11.12

$$|\det \mathbf{\Pi}| \leqslant \prod_{i=1}^{n} (|\varpi_{i,i}| + \Lambda_i(\mathbf{\Pi})) \leqslant (2n2^\rho)^n \leqslant 2^{n(\rho + \log_2(2n))} ,$$

which gives $\varepsilon_1 \leqslant n\tau 2^{-\mu} + 2^{(n(\rho+\log_2(2n))-\tau)/2}$. Therefore given the constraints $\tau \geqslant n \cdot (\rho + \log_2(2n)) + 2\lambda$ and $\mu \geqslant \rho + \alpha + \lambda$ we have that $\varepsilon_1 = \mathsf{negl}(\lambda)$.

Now, using Lemma 11.10 we obtain that the distribution of $(\mathsf{params}, \mathbf{r})$ if $(\varepsilon_1 + \varepsilon_2)$-close to that of $(\mathsf{params}, D_{2^\mu \mathbf{\Pi}})$ for

$$\varepsilon_2 = 2^{-\mu}\Big(\|\mathbf{r^{(1)}}\|_\infty \cdot \|\mathbf{\Pi}^{-1}\|_\infty + 1\Big) .$$

We consider the initial level-0 encoding c and write $c \equiv r_j^{(c)} \cdot g_j + m_j \bmod p_j$ and $y \equiv r_j^{(y)} \cdot g_j + 1 \bmod p_j$, and define

$$\mathbf{r^{(c)}} = \left(r_1^{(c)}, \ldots, r_n^{(c)}\right)^t \quad \text{and} \quad \mathbf{r^{(y)}} = \left(r_1^{(y)}, \ldots, r_n^{(y)}\right)^t .$$

Since $c_1 = c \cdot y \bmod x_0$, we have

$$\|\mathbf{r^{(1)}}\|_\infty = \left\|\left(r_j^{(c)} + r_j^{(c)} r_j^{(y)} g_j + m_j r_j^{(y)}\right)_{j=1,\ldots,n}\right\|_\infty \leqslant \|\mathbf{r^{(c)}}\|_\infty \cdot \|\mathbf{r^{(y)}}\|_\infty \cdot 2^{\alpha+2} .$$

Therefore $\|\mathbf{r}^{(\mathbf{1})}\|_\infty \leqslant 2^{2\rho+\log_2(\ell)+\alpha+2}$. Now, by Lemma 11.11, we have

$$\|\mathbf{\Pi}^{-1}\|_\infty \leqslant \frac{1}{\min_{i=1,\ldots,n}(|\varpi_{i,i}| - \Lambda_i(\mathbf{\Pi}))} \leqslant \frac{1}{n2^\rho - (n-1)2^\rho} \leqslant 2^{-\rho}\,.$$

This gives $\varepsilon_2 \leqslant 2^{-\mu} + 2^{\rho+\log_2(\ell)+\alpha+2-\mu}$. With the constraint $\mu \geqslant \rho+\alpha+\lambda$, Lemma 11.7 is proved. □

11.5 Attacks against our Multilinear Maps Scheme

11.5.1 Lattice Attack on the Encodings

We first describe a lattice attack against level-k encodings for all $k \in \mathbb{Z}$, $k \geqslant 0$. Consider an element $x_0 = \prod_{i=1}^n p_i$ and a set of τ integers $x_j \in \mathbb{Z}_{x_0}$ such that:

$$x_j \bmod p_i = r_{ij} g_i / z^k \bmod p_i\,,$$

where $r_{ij} \in (-2^\rho, 2^\rho) \cap \mathbb{Z}$ and $z \in [0, x_0)$. We want to estimate the complexity of the classical orthogonal lattice attack for recovering (some of) the noise values $r_{ij}g_i$.

This attack works by considering the integer vector formed by a subset of the x_j's, say $\mathbf{x} = (x_j)_{1\leqslant j\leqslant t}$ for some $n < t \leqslant \tau$, and relating the lattice of vectors orthogonal to $\mathbf{x}$ mod x_0 to the lattice of vectors orthogonal to each of the corresponding noise value vectors $\mathbf{r}_i = (r_{ij}g_i)_{1\leqslant j\leqslant t}$.

This attack is similar to the orthogonal lattice attack described in Section 7.2.4. In particular, let $L \subset \mathbb{Z}^t$ be the orthogonal lattice to $\mathbf{x}$ modulo x_0. A vector $\mathbf{u} \in L$ satisfies $\mathbf{u}\cdot\mathbf{x} \equiv 0 \pmod{x_0}$, so for each $i \in \{1,\ldots,n\}$, reducing modulo p_i gives:

$$\mathbf{u}\cdot\mathbf{r}_i \equiv 0 \pmod{p_i}.$$

In particular, if $\mathbf{u}$ is short enough to satisfy $\|\mathbf{u}\|\cdot\|\mathbf{r}_i\| < p_i$, this implies $\mathbf{u}\cdot\mathbf{r}_i = 0$ in $\mathbb{Z}$. Overall we get a time complexity of $2^{\Omega(\gamma/\eta^2)}$, similar as in Chapter 7.

11.5.2 GCD Attack on the Zero-testing Parameter

We consider the ratio modulo x_0 of two coefficients from the zero-testing vector $\mathbf{p}_{zt}$, namely $u := (\mathbf{p}_{zt})_1/(\mathbf{p}_{zt})_2 \bmod x_0$. From Equation (11.8) we obtain for all $1 \leqslant i \leqslant \ell$:

$$u \equiv h_{i1}/h_{i2} \pmod{p_i}$$

We can therefore recover p_i by computing $\gcd(h_{i2}\cdot u - h_{i1}, x_0)$ for all possible h_{i1}, h_{i2}. Since the h_{ij}'s are upper bounded by 2^β in absolute value, the attack has complexity $\mathcal{O}(2^{2\beta})$. By using a technique similar to [CN12], the attack complexity can be reduced to $\tilde{\mathcal{O}}(2^\beta)$.

11.5.3 Hidden Subset Sum Attack on Zero Testing

One can consider an attack similar to Section 11.5.1 on the zero-testing parameters. The zero-testing vector $\boldsymbol{\omega} = c\cdot\mathbf{p}_{zt} \bmod x_0$ corresponding to an encoding c of zero can be written as:

$$\boldsymbol{\omega} = \sum_{i=1}^n \Big(r_i \cdot \prod_{i'\neq i} p_{i'}\Big)\mathbf{h}_i = \sum_{i=1}^n R_i\mathbf{h}_i,$$

where the $\mathbf{h}_i$'s are the lines of the zero-testing matrix $\mathbf{H}$. This gives an expression of $\boldsymbol{\omega}$ as a linear combination of the unknown vectors $\mathbf{h}_i$, so we can think of an approach similar to the hidden subset sum attack of Nguyen and Stern [NS99] to recover the unknown $\mathbf{h}_i$'s or the R_i's.

Such an approach, like the one from the previous section, would first involve computing the lattice $L \subset \mathbb{Z}^n$ of vectors $\mathbf{u}$ orthogonal to $\boldsymbol{\omega}$ modulo x_0, and hoping that the vectors in a reduced basis of L are short enough that they must necessarily be orthogonal to each of the $\mathbf{h}_i$'s in $\mathbb{Z}^n$.

However, L is a lattice of rank n and volume $x_0 \approx 2^{n\cdot\eta}$, so we expect its shortest vectors to be of length roughly 2^η. The attack can then only work if such a short vector $\mathbf{u}$ is necessarily orthogonal

to each of the $\mathbf{h}_i$'s in $\mathbb{Z}^n$. Equivalently, this will happen if the vector $\mathbf{v} = (\mathbf{u} \cdot \mathbf{h}_1, \ldots, \mathbf{u} \cdot \mathbf{h}_n) \in \mathbb{Z}^n$, which we know is orthogonal to $(R_1, \ldots, R_n)$ modulo x_0, is significantly shorter than the shortest vector in the lattice L' of vectors orthogonal to $(R_1, \ldots, R_n)$ modulo x_0. But again L' is of rank n and determinant x_0, so its shortest vectors is of length about 2^η, and hence $\mathbf{v}$ will typically not be shorter.

Therefore, the Nguyen-Stern hidden subset sum attack does not apply to our setting.

11.5.4 Attacks on the Inverse Zero Testing Matrix

A related observation is that if $\boldsymbol{\omega} = c \cdot \mathbf{p}_{zt} \bmod x_0$ denotes again the zero-testing vector corresponding to an encoding c of zero, and $\mathbf{T} = (\mathbf{H}^{-1})^t$ is the inverse zero-testing matrix, then, by Equation (11.10):

$$\mathbf{T}\boldsymbol{\omega} = (R_1, \ldots, R_n) \in \mathbb{Z}^n,$$

where, as above, $R_i = r_i \cdot (x_0/p_i)$. In particular, if the lines of $\mathbf{T}$ are written as $\mathbf{t}_i$, we get that for each $i \in \{1, \ldots, n\}$:

$$p_i \mathbf{t}_i \cdot \boldsymbol{\omega} \equiv 0 \pmod{x_0}.$$

Thus, since $p_i\mathbf{t}_i$ is relatively small, we might hope to recover it as a short vector in the lattice of vectors orthogonal to $\boldsymbol{\omega}$ modulo x_0. However, as we have seen previously, we expect a reduced basis of that lattice to have vectors of length roughly 2^η, whereas $p_i\mathbf{t}_i$ is of length about $2^{\eta+\beta}$. By the Gaussian heuristic, even when $\beta = \mathcal{O}(1)$, there are exponentially many linear combinations of the reduced basis vectors under the target length, and hence we cannot hope to recover $p_i\mathbf{t}_i$ in that fashion.

A more sophisticated attack based on the same observation uses the result of Hermann and May [HM08] on solving linear equations modulo an unknown factor of a public modulus. In our case, the components t_{ij} of $\mathbf{t}_i$ form a small solution (smaller than 2^β) of the linear equation:

$$\sum_{j=1}^{n} \omega_j \cdot t_{ij} \equiv 0 \pmod{x_0/p_i}$$

modulo the unknown factor x_0/p_i of x_0. The technique of Hermann–May can thus recover $\mathbf{t}_i$ and factor x_0 provided that β is small enough relative to x_0. For sufficiently large n, by [HM08, Theorem 4], this should be possible as long as $\beta/\eta < 1$. However, as noted by the authors, the complexity of the attack is exponential in n (it involves reducing a lattice of dimension $\Omega(\exp n)$), and there does not seem to be a way to approximate the solution in polynomial time, so the attack does not apply to our setting even though we do choose $\beta < \eta$.

11.5.5 A Note on GGH's Zeroizing Attack

In [GGH13a] the authors describe a "zeroizing" attack against their scheme that consists in multiplying a given level-i encoding c by a level-$(\kappa - i)$ encoding of 0 to get a level-κ encoding of 0, and then multiplying by the zero-testing parameter $\mathbf{p}_{zt}$; one obtains an encoding of a (deterministic) multiple of the coset of c but in the plaintext space. This attack does not enable to solve the GDDH problem because one does not get a small representative of that coset, but it enables to solve some decisional problems involving low-level (below κ) encodings, such as the decisional subgroup membership problem using composite-order maps, and the decisional linear problem.

Surprisingly this attack does not seem to apply against our scheme; namely we do not get a similar encoding in the plaintext space from the zero-testing parameter. Therefore the subgroup membership assumption and the decision linear assumption (DLIN) could still hold in our scheme.

Remark 11.13. Note that to instantiate their verifier-based password-authenticated key exchange based on multilinear maps [BP13], Benhamouda and Pointcheval cannot use the multilinear maps scheme [GGH13a] because of this attack. Our scheme is therefore *currently* the only scheme that can underly these constructions.

11.6 Conclusion

In this chapter, we proposed a new approximate multilinear maps scheme based on Garg, Gentry and Halevi's framework [GGH13a], but relying on different techniques and assumptions. Our hardness assumptions are new, as the ones in [GGH13a]; we sustain them by a careful study of attacks applicable to our scheme, and we provide asymptotic parameters constraints. As a result of independent interest, we stated a "Leftover Hash Lemma over lattices", that is at the core of our rerandomization procedure. This new technique is directly applicable to the scheme of [GGH13a].

Due to the novelty of the results, even though most of the security analysis is similar to the security analysis of the DGHV homomorphic encryption scheme extensively studied in Part II of this thesis, cryptanalysis is *much much* needed in the field to assess the reality of this exciting primitive.

A lot of open problems remain; notably to design multilinear maps based on more "classical" assumptions such as LWE. Even though our scheme, and the one of [GGH13a], appear not to be very practical, it is a very exciting problem to optimize the current primitives, in order to make them practical enough to be used for real life applications. An optimization of the GGH scheme, called GGHLite, has very recently been proposed by Langlois, Stehlé and Steinfield [LSS14], and might yield interesting practical results. In Chapter 12, we optimize upon the scheme of this chapter and present the first implementation of (approximate) cryptographic multilinear maps.

11.A Uniform Sampling of a Parallelepiped

In order to sample a uniformly random element in the half-open parallelepiped defined by the column vectors ϖ_j of matrix $\mathbf{\Pi} \in \mathbb{Z}^{n\times n}$, one can proceed as follows.

First, compute the Smith Normal Form for $\mathbf{\Pi}$. This is easily done in polynomial time, and can be done with near optimal complexity using Storjohann's algorithm [Sto96]. This yields a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of $\mathbb{Z}^n$ and positive integers $d_1, \ldots, d_n$ such that $(d_1\mathbf{b}_1, \ldots, d_n\mathbf{b}_n)$ is a basis of the lattice L generated by the columns of $\mathbf{\Pi}$.

Now if we pick integers $x_1, \ldots, x_n$ at random such that x_i is uniformly distributed in $\{0, \ldots, d_i - 1\}$, then clearly, the vector $\mathbf{x} = x_1\mathbf{b}_1 + \cdots + x_n\mathbf{b}_n$ is uniformly distributed modulo L.

To get a uniformly distributed vector in the half-open parallelepiped defined by the ϖ_j's, it then suffices to apply Babai's round-off algorithm [Bab86], *i.e.* write $\mathbf{x}$ as a rational linear combination $\xi_1\varpi_1 + \cdots + \xi_n\varpi_n$ of the ϖ_j's and return the vector $\mathbf{y}$ given by:

$$\mathbf{y} = \sum_{j=1}^{n} \big(\xi_j - \lfloor \xi_j \rfloor\big)\varpi_j.$$

That vector is congruent to $\mathbf{x}$ modulo L, so it is also in $\mathbb{Z}^n$ and uniformly distributed modulo L, and it belongs to the parallelepiped by construction, so it is indeed a uniformly distributed element of the parallelepiped.

11.B Generation of the Matrix H

For the construction of zero-testing parameters, we need to pick, with sufficient entropy, an invertible matrix $\mathbf{H} \in \mathbb{Z}^{n\times n}$ in such a way that both its operator norm and the norm of its inverse are not too large, namely $\|\mathbf{H}\|_\infty \leqslant 2^\beta$ and $\|\mathbf{H}^{-1}\|_\infty \leqslant 2^\beta$. In Section 11.3 the bounds must actually hold for $\mathbf{H}^t$, so we take the transpose of the resulting matrix.

For that purpose, we propose the following approach. For any matrix A of size $\lfloor n/2 \rfloor \times \lceil n/2 \rceil$ with coefficients in $\{-1, 0, 1\}$, define $\mathbf{H}_A \in \mathbb{Z}^{n\times n}$ as:

$$\mathbf{H}_A = \begin{pmatrix} I_{\lfloor n/2 \rfloor} & A \\ 0 & I_{\lceil n/2 \rceil} \end{pmatrix}.$$

Each line of $\mathbf{H}_A$ has at most $1 + \lceil n/2 \rceil$ non zero coefficients, each in $\{-1, 0, 1\}$, so we clearly have $\|\mathbf{H}_A\|_\infty \leqslant 1 + \lceil n/2 \rceil$. Moreover, $\mathbf{H}_A$ is invertible with $\mathbf{H}_A^{-1} = \mathbf{H}_{(-A)}$, so that the operator norm of its inverse admits a similar bound.

Similarly, the transpose $\mathbf{H}'_A$ of $\mathbf{H}_A$ also satisfies $\|\mathbf{H}'_A\|_\infty \leqslant 1 + \lceil n/2 \rceil$ (in fact, the slightly better bound by $1 + \lfloor n/2 \rfloor$ also holds) and so does its inverse.

Now let:

$$\beta' = \left\lfloor \frac{\beta}{\lceil \log_2(1 + \lceil n/2 \rceil) \rceil} \right\rfloor,$$

and generate β' uniformly random matrices A_i of size $\lfloor n/2 \rfloor \times \lceil n/2 \rceil$ with coefficients in $\{-1, 0, 1\}$; then pick $\mathbf{H}_i$ randomly as either $\mathbf{H}_{A_i}$ or its transpose for each $i \in \{1, \ldots, \beta'\}$, and finally compute $\mathbf{H}$ as the product of the $\mathbf{H}_i$'s. Then, since operator norms are sub-multiplicative, we have:

$$\|\mathbf{H}\|_\infty \leqslant \prod_{i=1}^{\beta'} \|\mathbf{H}_i\|_\infty \leqslant (1 + \lceil n/2 \rceil)^{\beta'} \leqslant 2^\beta,$$

and $\mathbf{H}^{-1}$ satisfies the same bound. The set of matrices $\mathbf{H}$ obtained in this manner is not very simple to describe but it is exponentially large.

Chapter 12

Implementation of a $N \geqslant 3$-partite Diffie-Hellman Key Exchange

12.1 Introduction

We view the construction of approximate multilinear maps in [GGH13a] as a significant breakthrough. Therefore we find it interesting to obtain a new scheme based on different techniques (even relatively similar) as in Chapter 11, and this provides more confidence in the feasibility of such a construction. Also since the basic schemes of Chapter 11 and of Garg, Gentry and Halevi appear to be rather unpractical, it is interesting to find optimizations (even completely heuristic) to obtain a scheme that can be implemented in practice. In this chapter, we describe the first implementation of cryptographic (approximate) multilinear maps, and benchmarking results for multipartite Diffie-Hellman key exchanges. We obtained that an optimized variant of the construction of Chapter 11 is arguably practical as secure one-round key exchanges up to 7 users run in a few seconds on a mid-range computer.

This chapter includes the implementation section of the article Practical Multilinear Maps over the Integers [CLT13b], cosigned with J.-S. Coron and M. Tibouchi, and published at Crypto 2013 [CG13a]. The full version of the article is available at [CLT13c], and the proof-of-concept implementation of our scheme is openly available at [Lep13]. Additionally, we present two works in progress as appendices. In Appendix 12.A, we propose a quadratic sampling algorithm that allows one to *provably* reduce the size of the public parameters (contrary to our heuristic implemented optimization). In Appendix 12.B, we provide a min-entropy/memory trade-off by reducing the number of elements in the zero-testing vector $\mathbf{p}_{zt}$.

Multilinear Maps and Diffie-Hellman Key Exchange. In [Jou00], Joux proposed to use cryptographic bilinear maps to perform a one-round tripartite Diffie-Hellman key exchange, that is a protocol that enables three parties to share a common secret without exchanging any messages. In 2003, Boneh and Silverberg generalized this protocol to multiple parties assuming the existence of cryptographic multilinear maps, but let as a challenging open problem to build the required multilinear maps [BS03].

The candidate approximate multilinear maps scheme of Garg, Gentry and Halevi differs quite substantially from the "ideal" multilinear maps envisaged by Boneh and Silverberg. However, since the multilinear analogue of the decisional Diffie-Hellman (the Graded Decisional Diffie-Hellman problem, *cf.* Section 11.2.4) seems to be hard in their setting, that yields some hope for meaningful applications.

In particular, they show in [GGH13a] that the "obvious" N-partite Diffie-Hellman key exchange protocol can be instantiated by their approximation of a $(N-1)$-linear maps, and proven to be secure under the GDDH assumption.

Our Contribution. In this chapter, we show that similarly to GGH [GGH13a], our candidate multilinear maps of Chapter 11 is applicable to the multipartite Diffie-Hellman key exchange protocol. Our contribution is to describe the first implementation of cryptographic multilinear maps.

It appears that the basic versions of both [GGH13a] and our scheme as described in Chapter 11 are rather unpractical, because of the huge public parameter size required to randomize the encodings. Therefore we use a simple optimization that consists in storing only a small subset of the public elements and combining them pairwise to generate the full public-key. Such optimization was originally described in [GH11b] for reducing the size of the encryption of the secret-key bits in the implementation of Gentry's FHE scheme [Gen09]. It was also used in [CMNT11] to reduce the public-key size of the DGHV scheme; however, as opposed to the latter work our randomization of encodings is heuristic only, whereas in [CMNT11] the semantic security was still guaranteed. Thanks to this optimization our construction becomes relatively practical: for reasonable security parameters a multipartite Diffie-Hellman computation with 7 users (resp. 26 users) requires less than 40 seconds (resp. 5 minutes), with a public parameter size of roughly 2.6 GBytes (resp. 8.3 GBytes); a proof-of-concept implementation is openly available at [Lep13].

In appendix, we describe an optimization similar to the pairwise combination of public key parameters [GH11b, CMNT11] for the sampling algorithm (rather than the randomization algorithm). We also describe a technique to reduce the number of elements of the zero-testing vector, providing a min-entropy/memory trade-off for implementations.

12.2 Diffie-Hellman One-Round Key Exchange

In the seminal paper [DH76], Diffie and Hellman introduced a non-interactive (*i.e.* one round) key exchange protocol which is still nowadays one of the most famous cryptographic primitives. This protocol enables two parties, say Alice and Bob, to share a common secret without exchanging any messages. We recall the protocol as stated in the introduction (Chapter 2).

The parameters consist of a cyclic group $\mathbb{G}$ (denoted additively) of prime order p, generated by $g \in \mathbb{G}$. Alice (resp. Bob) generates a key pair $(\mathsf{sk}_A, \mathsf{pk}_A) = (x, x \cdot g)$ (resp. $(\mathsf{sk}_B, \mathsf{pk}_B) = (y, y \cdot g)$) where $x \leftarrow \mathbb{Z}_p$ (resp. $y \leftarrow \mathbb{Z}_p$) and makes her (resp. his) public key openly available. When Alice and Bob want to share a secret, they both compute a shared secret key $(xy) \cdot g = y \cdot (x \cdot g) = x \cdot (y \cdot g)$ with their own secret key and the other's public key.

This incredibly simple protocol (when hashing the resulting secret along with both identities) fulfills all the properties expected by a non-interactive key exchange protocol and is very efficient. Its security is based on the Decisional Diffie-Hellman problem, in which one has to distinguish the distributions $(a \cdot g, b \cdot g, (ab) \cdot g)$ and $(a \cdot g, b \cdot g, c \cdot g)$ for random $a, b, c \in \mathbb{Z}_p$ where g generates a cyclic group $\mathbb{G}$ of prime order p.

12.2.1 Tripartite Diffie-Hellman Key Exchange

In 2000, Joux proposed a tripartite generalization of Diffie-Hellman key exchange protocol using the Weil and Tate pairings [Jou00], still non-interactive.

In particular, it uses a symmetric bilinear map, that is a map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p, which is bilinear (*i.e.* for all $g, h \in \mathbb{G}$ and $a \in \mathbb{Z}_p$, $e(a \cdot g, h) = a \cdot e(g, h) = e(g, a \cdot h)$) and non-degenerate (*i.e.* if $\mathbb{G} = (g)$, then $\mathbb{G}_T = (e(g, g))$).

Now assume that three parties, say Alice, Bob and Carroll, want to share a secret key. Alice (resp. Bob, resp. Carroll) generates a key pair $(\mathsf{sk}_A, \mathsf{pk}_A) = (x, x \cdot g)$ (resp. $(\mathsf{sk}_B, \mathsf{pk}_B) = (y, y \cdot g)$, resp. $(\mathsf{sk}_C, \mathsf{pk}_C) = (z, z \cdot g)$) and publishes her (resp. his, resp. her) public key. The shared secret is the value

$$(xyz) \cdot e(x, x) = z \cdot e(x \cdot g, y \cdot g) = y \cdot e(x \cdot g, z \cdot g) = x \cdot e(y \cdot g, z \cdot g)\,.$$

The security of this protocol relies on the Bilinear Decisional Diffie Hellman problem, in which one has to distinguish the distributions $(a \cdot g, b \cdot g, c \cdot g, (abc) \cdot e(g, g))$ and $(a \cdot g, b \cdot g, c \cdot g, d \cdot e(g, g))$ for random $a, b, c, d \in \mathbb{Z}_p$, where g generates a cyclic group $\mathbb{G}$ of prime order p and $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a cryptographic bilinear map.

12.2.2 N-partite Diffie-Hellman Key Exchange

In 2003, Boneh and Silverberg generalized this result and showed how to perform a multipartite Diffie-Hellman key exchange between N users assuming the existence of cryptographic $(N-1)$-linear

maps [BS03]. Consider N parties wishing to set up a shared secret key s using a one-round protocol (*i.e.* in which each party broadcasts one value to all other parties and the N broadcasts occur simultaneously). Once the N broadcast values are known, each party should be able to locally compute a global shared secret s. Let us recall the definition of such a protocol using the notation from [BS03, GGH13a]. A one-round N-way key exchange scheme consists of the following three randomized probabilistic polynomial-time algorithms:

$\mathsf{Setup}(\lambda, N)$. From a security parameter λ and the number of participants N, this algorithm runs in polynomial time in λ, N and outputs public parameters params.

$\mathsf{Publish}(\mathsf{params}, i)$. Given a value $i \in \{1, \ldots, N\}$, this algorithm outputs a key pair $(\mathsf{pub}_i, \mathsf{priv}_i)$. Party i broadcasts pub_i to all other parties and keep priv_i secret.

$\mathsf{KeyGen}(\mathsf{params}, i, \mathsf{priv}_i, \{\mathsf{pub}_j\}_{j\neq i})$. Party i computes KeyGen on all the collected public (broadcast) values $\{\mathsf{pub}_j\}_{j\neq i}$ and its secret value priv_i. This algorithm outputs a key s_i.

The protocol is said to be correct if the N parties generate the same shared key s with high probability, *i.e.* $s = s_1 = \cdots = s_N$. A correct protocol is said to be secure if, given all N public values pub_i, no polynomial time algorithm can distinguish the true shared secret s from a random string.

Assume we have a cryptographic $(N-1)$-linear map $e\colon \mathbb{G}^{N-1} \to \mathbb{G}_T$. Each user $i \in \{1, \ldots, N\}$ generates a key pair $(\mathsf{sk}_i, \mathsf{pk}_i) = (x_i, x_i \cdot g)$ and publishes the public key. With its secret key and the others' public key, each user can generate the shared secret value

$$\Big(\prod_{i=1}^{N} x_i\Big) \cdot e(g, \ldots, g) = x_1 \cdot e(x_2 \cdot g, \ldots, x_N \cdot g) = \cdots = x_N \cdot e(x_1 \cdot g, \ldots, x_{N-1} \cdot g)\,.$$

This protocol is secure under the Multilinear Decisional Diffie-Hellman problem (*cf.* Definition 11.1) which aims, as previously, to distinguish an encoding of a legitimate product from a random encoding.

12.3 N-partite Diffie-Hellman Key Exchange Using Approximate Multilinear-Maps

As pointed out in Chapter 11, the only cryptographic multilinear maps currently available are only approximations of the cryptographic multilinear maps of Boneh and Silverberg. However, these approximations are sufficient to perform a one-round N-partite Diffie-Hellman key exchange protocol in the common reference string model (because the public parameters hide some secrets – namely the p_i's and z), under the GDDH assumption with $N = \kappa + 1$ users.

Our construction is the same as in [GGH13a]. Consider N parties wishing to set up a shared secret key s using a one-round protocol (*i.e.* each party broadcasts one value to all other parties). The protocol is as follows:

$\mathsf{Setup}(1^\lambda, 1^N)$. Output $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$ as the public parameter, with $\kappa = N - 1$.

$\mathsf{Publish}(\mathsf{params}, i)$. Each party i samples a random $c_i \leftarrow \mathsf{samp}(\mathsf{params})$ as a secret key, and publishes as the public key the corresponding level-1 encoding

$$c'_i \leftarrow \mathsf{reRand}(\mathsf{params}, 1, \mathsf{enc}(\mathsf{params}, 1, c_i))\,.$$

$\mathsf{KeyGen}(\mathsf{params}, \mathbf{p}_{zt}, i, c_i, \{c'_j\}_{j\neq i})$. Each party i computes $\tilde{c}_i = c_i \cdot \prod_{j\neq i} c'_j$, and uses the extraction routine to compute the (shared) key $s \leftarrow \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \tilde{c}_i)$.

The correctness of the protocols follows from the fact that all parties get valid encodings of the same vector, hence with the parameters as given in Section 11.3.1, the extraction property implies that they should extract the same key with overwhelming probability.

The security of the protocol follows from the randomness property of the extraction procedure and the GDDH hardness assumption.

Theorem 12.1 ([GGH13a]). *The protocol described above is a secure one-round N-way Diffie-Hellman key exchange protocol if the GDDH assumption holds for the underlying encoding scheme.*

Proof. We need to show that an attacker who sees all the public keys cannot distinguish the output of the first party (say) from a uniform random string. Now Party 1 extracts the same string as one would extract from $c = \mathsf{reRand}(\mathsf{params}, \kappa, \prod_{i=1}^{N} + c_i)$.

By GDDH, the adversary cannot distinguish c from $c' = \mathsf{reRand}(\mathsf{params}, \kappa, b)$ for a random and independent $b \leftarrow \mathsf{samp}(\mathsf{params})$. Now by the randomness property of the sampling procedure (*i.e.* Lemma 11.6), b is nearly uniformly distributed in R. Therefore, by the randomness property of the extraction function, we conclude that $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, c')$ is a nearly uniform string, completing the proof. □

12.4 Optimizations and Implementation

In this section we describe an implementation of our scheme in the one-round N-way Diffie-Hellman key exchange protocol (*cf.* Section 12.3).

We note that without optimizations the size of the public parameters makes the scheme of Chapter 11 completely unpractical; this is also the case in [GGH13a]. Namely, for sampling we need to store at least $n \cdot \alpha$ encodings (resp. $n \cdot \rho$ encodings for re-randomization), each of size $n \cdot \eta$ bits; the public-key size is then at least $n^2 \cdot \eta \cdot \alpha$ bits. With $n \simeq 10^4$, $\eta \simeq 10^3$ and $\alpha \simeq 80$, the public-key size would be at least 1 TB.[1] Therefore we use three heuristic optimizations to reduce the memory requirement.

1. Non-uniform sampling: for the sampling algorithm we use a small number of encodings ℓ only; this implies that the sampling cannot be proved uniform anymore.

2. Quadratic re-randomization: we only store a small subset of encodings which are later combined pairwise to generate the full set of encodings. This implies that the randomization of encodings becomes heuristic only.

3. Integer p_{zt}: we use a single integer p_{zt} instead of a vector $\mathbf{p}_{zt}$ with n components. An encoding c of zero still gives a small integer $\omega = p_{zt} \cdot c \bmod x_0$, but the converse does not necessarily hold anymore.

12.4.1 Non-uniform Sampling

For sampling level-zero encodings we use a smaller value for ℓ, the number of encodings x_j in the public parameters. There is a simple meet-in-the-middle attack with complexity $\mathcal{O}(2^{\ell/2})$; therefore we take $\ell = 2\lambda$. In this case the sampling cannot be proved uniform in $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ anymore. However this does not seem to make the GDDH problem easier.

Note also that for such small ℓ given a level-0 encoding c, one can efficiently recover the coefficients of the subset sum with LLL, since this is a subset-sum problem with density $\ell/(\eta \cdot n) \ll 1$; however this does not give an attack, as in GDDH such level-0 encoding c is not available.[2]

12.4.2 Quadratic Re-randomization

To reduce the parameters size we use a simple optimization that consists in storing only a small subset of the public elements and combining them pairwise to generate the full public-key. Such optimization was originally described in [GH11b] for reducing the size of the encryption of the secret-key bits in the implementation of Gentry's FHE scheme [Gen09]. It was also used in [CMNT11] to reduce the public-key size of the DGHV scheme; however, as opposed to [CMNT11] our randomization of encodings becomes heuristic only, whereas in [CMNT11] the semantic security was still guaranteed.

[1] In [GGH13a] the following approximate setting is suggested: $n = \tilde{\mathcal{O}}(\kappa\lambda^2)$, $q = 2^{n/\lambda}$ and $m = \mathcal{O}(n^2)$. The public-key size contains at least m encodings of size $n \log_2 q$ bits each. Taking exactly $n = \kappa\lambda^2$ and $m = n^2$, the public-key size is then $m \cdot n \cdot (n/\lambda) = n^4/\lambda = \kappa^4\lambda^7$. With $\kappa = 6$ and $\lambda = 80$, we get a public-key size of 3400 TB.

[2] Alternatively, as described in Appendix 12.A, one could use the same quadratic technique as in [CMNT11]; in that case the sampling could still be proved uniform in R.

For re-randomization we only store $\Delta = \lfloor \sqrt{n} \rfloor$ encoding $x_j^{(0)}$ at level 0 and also Δ encodings $x_j^{(1)}$ at level 1. The $x_j^{(0)}$ encode random $\mathbf{m}_j \in R$, while the $x_j^{(1)}$ are encodings of $\mathbf{0}$. Then by pairwise multiplication we can generate $\Delta^2 \simeq n$ randomization elements at level 1, which are all encodings of $\mathbf{0}$. More precisely, we have for $b = 0, 1$ and $1 \leqslant j \leqslant \Delta$:

$$x_j^{(b)} \equiv \frac{r_{ij}^{(b)} \cdot g_i + (1-b) \cdot f_{ij}}{z^b} \pmod{p_i},$$

where $r_{ij}^{(b)}$ are random ρ-bit integers, and f_{ij} are random integers modulo g_i.

Given a level-1 encoding c_1, we randomize it using a random subset-sum of pairwise products of the previous encodings:

$$c_1' = c_1 + \sum_{i,j=1}^{\Delta} \alpha_{ij} \cdot x_i^{(0)} \cdot x_j^{(1)} \bmod x_0,$$

where the α_{ij}'s are random bits; note that we don't use the encodings Π_j anymore. As a further optimization, we can use as in [CMNT11] a sparse vector α_{ij}, with small Hamming weight θ. There is a meet-in-the-middle attack of complexity $\mathcal{O}(n^{\theta/2})$. In our implementation we take $\theta = 16$; the reRand operation then becomes very efficient.

Writing as previously $c_1' \equiv (r_i' \cdot g_i + m_i)/z \pmod{p_i}$, we obtain under this optimization:

$$|r_i' \cdot g_i + m_i| \leqslant (\ell + \theta) \cdot 2^{2(\rho+\alpha)}.$$

When computing the product of κ such level-1 encodings and one level-0 encoding as in multipartite Diffie-Hellman key exchange, we obtain the following updated bound for the $\log_2$ infinite norm of the vector $\mathbf{r}$ from Lemma 11.8:

$$\rho_f \leqslant \kappa \cdot (2\rho + 2\alpha + \log_2(\ell + \theta)) + \rho + \log_2 \ell + 1.$$

12.4.3 Zero-Testing Element

Instead of generating a zero-testing vector $\mathbf{p}_{zt}$ with n components, we publish a zero-testing element p_{zt} which is a single integer:

$$p_{zt} = \sum_{i=1}^{n} h_i \cdot (z^\kappa \cdot g_i^{-1} \bmod p_i) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0$$

where the h_i's are random β-bit integers. Therefore, we obtain a single integer $\omega = p_{zt} \cdot c \bmod x_0$, with

$$\omega = \sum_{i=1}^{n} h_i \cdot \big(r_i + m_i \cdot (g_i^{-1} \bmod p_i)\big) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0.$$

As before, if $\|\mathbf{r}\|_\infty < 2^{\rho_f}$ we still have $|\omega| < x_0 \cdot 2^{-\nu-\lambda-2}$. However the converse is no longer true: we can have $|\omega| < x_0 \cdot 2^{-\nu}$ for an encoding of a non-zero vector $\mathbf{m}$. This implies that two encodings of different vectors can now extract to the same value. While it is actually easy to generate such collisions using LLL, this does not seem to give an attack against the GDDH problem. The resulting scheme is therefore no longer computationally (and statistically) zero-test secure (*cf.* Definition 11.4).

12.5 Practical Results

We have implemented a one-round N-way Diffie-Hellman key exchange protocol with $N = 3, 5, 7$ and 26 users, in C++ using the GMP library to perform operations on large integers. We refer to Section 12.3 for a description of the protocol. Our proof-of-concept implementation is openly available for the community to reproduce our experiments at [Lep13].

We provide our concrete parameters and the resulting timings in Table 12.1 (on page 169), for security parameters ranging from 52 to 80 bits. We used algorithms similar to the algorithms used to derive parameters for the multi-slot DGHV scheme in Chapter 7.

The timings of Table 12.1 show that our scheme is relatively practical, as the KeyGen phase of the 7-partite (resp. 26-partite) Diffie-Hellman protocol requires only a few seconds (resp. minutes) per user; however the parameter size is still very large even with our optimizations.

Remark 12.2. We could not assess the practicality of our scheme for larger κ because of the costly Setup phase. However, we can easily estimate the timings more precisely for a multiplication and the rerandomization procedure (using random elements instead of well-formed ones).[3] In particular, we obtain that for $\kappa = 100$ levels each multiplication (resp. modular multiplication) takes about 10 seconds (resp. 30 seconds) and the rerandomization about 3 minutes. In particular, we can roughly estimate that the KeyGen procedure of a 101-partite key exchange takes about one hour per participant.

12.6 Conclusion

In this chapter, we proposed some heuristic optimizations for the approximate multilinear maps scheme over the integers described in Chapter 11, and we described the first implementation of cryptographic multilinear maps (openly available at [Lep13]).[4] In particular, we provide parameters and timings for a N-partite Diffie-Hellman Key Exchange protocol for $N = 3, 5, 7$ and 26.

Our results show that using cryptographic multilinear maps for a small number of levels might be practical enough for some applications, as a key exchange between seven users only took a few seconds per user with our proof-of-concept implementation on a mid-range computer.

The two constructions of approximate multilinear maps published in 2013 [GGH13a, CLT13b] provoked a wave of new results and constructions based on multilinear maps. Unfortunately, all these constructions require a large number of multilinearity levels, practically unreachable by our implementation. It remains very challenging to propose optimizations to the approximate multilinear maps scheme candidates, or to propose a new scheme, in order to handle a large number of levels in a reasonable time.

12.A Quadratic Sampling for Level-Zero Encodings

In order to reduce the size of the public parameters params, one could use the same quadratic encryption technique as in [CMNT11]. The idea consists in combining on-the-fly a smaller subset of public parameters x'_j's multiplicatively. In the following, we describe our new technique to sample level-zero encodings and we show that the sampling can still be proved uniform in R.

Quadratic Sampling. $c \leftarrow \mathsf{quadsamp}(\mathsf{params})$. Similarly as before, we publish a set of 2ℓ integers $x'_{j,b}$ for $1 \leqslant j \leqslant \ell$ and $b \in \{0, 1\}$, each one being a level-0 encoding of a random message $\mathbf{a}_{j,b} = (a_{ij,b})_{i=1}^{n}$ in R. We denote

$$1 \leqslant j \leqslant \ell, b \in \{0, 1\}, \qquad x'_{j,b} \equiv r'_{ij,b} \cdot g_i + a_{ij,b} \pmod{p_i} \tag{12.1}$$

with $r'_{ij,b}$'s randomly generated in $(-2^\rho, 2^\rho) \cap \mathbb{Z}$.

Our *quadratic* randomized sampling algorithm quadsamp(params) works as follows: we generate a random ς-bit integer vector $\mathbf{b} = (b_{ij})$ of size ℓ^2 and output the level-0 encoding

$$c = \sum_{1 \leqslant i,j \leqslant \ell} b_{ij} \cdot x'_{i,0} \cdot x'_{j,1} \mod x_0 \,.$$

The output c is a level-0 encoding:

$$c \equiv r_i \cdot g_i + m_i \pmod{p_i}$$

of some vector $\mathbf{m} \in \mathbb{Z}^n$; for such level-0 encodings we get $|r_i \cdot g_i + m_i| \leqslant \ell^2 \cdot 2^{\varsigma+2\rho+2\alpha}$ for all i.

The following Lemma states that, as required, the distribution of $\mathbf{m}$ is statistically close to uniform over $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$; the proof is based on applying the Leftover Hash Lemma over the set R (*cf.* Section 3.3.1).

[3] It is worth noting that one does not need the whole public key when multiplying (but only x_0) nor during the rerandomization procedure (only x_0 and the rerandomization elements).

[4] Note that, to this date, *no* other implementation of multilinear maps has been proposed in the literature, even though the GGH scheme has been improved upon in [LSS14].

Lemma 12.3. *Let $c \leftarrow \mathsf{quadsamp}(\mathsf{params})$ and write $c \equiv r_i \cdot g_i + m_i \pmod{p_i}$. Let $\varsigma \cdot \ell^2 \geqslant \max\left(n\alpha, (2\varsigma+1)\ell + \log_2(\ell^2) + \log_2(n)\right) + 2\lambda$. The distribution of $(\mathsf{params}, \mathbf{m})$ is statistically close to the distribution of $(\mathsf{params}, \mathbf{m}')$ where $\mathbf{m}' \leftarrow R$.*

ϵ-Pairwise Independent Hash Functions Family. We first recall the notion of ε-pairwise independence introduced in [CMNT11].

Definition 12.4. A family $\mathcal{H}$ of hash functions $h\colon X \to Y$ is ε-pairwise independent if

$$\sum_{x \neq x'} \left(\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(x')] - \frac{1}{|Y|} \right) \leqslant |X|^2 \cdot \frac{\varepsilon}{|Y|}.$$

The following generalization of the usual Leftover Hash Lemma is proved in [CMNT11].

Lemma 12.5 (Leftover hash lemma)**.** *Let $\mathcal{H}$ be a family of ε-pairwise independent hash functions. Suppose that $h \leftarrow \mathcal{H}$ and $x \leftarrow X$ are chosen uniformly and independently. Then $(h, h(x))$ is $(\frac{1}{2}\sqrt{|Y|/|X| + \varepsilon})$-uniform over $\mathcal{H} \times Y$.*

Let $\mathcal{H}$ be a hash family from $X = \{0, \ldots, 2^\varsigma - 1\}^{\ell^2}$ associated to elements $\mathbf{a}_0 = (a_{i,0})_i$ and $\mathbf{a}_1 = (a_{i,1})_i$ of R for $1 \leqslant i \leqslant n$. For $\mathbf{b} \in \{0, \ldots, 2^\varsigma - 1\}^{\ell^2}$, we let:

$$h(\mathbf{b}) = \sum_{1 \leqslant i,j \leqslant \ell} b_{ij} \cdot \mathbf{a}_0 \cdot \mathbf{a}_1$$

where the multiplication is component-wise in R.

Lemma 12.6. *The hash function family $\mathcal{H}$ is ϵ-pairwise independent, with*

$$\epsilon = \frac{1}{\min(g_i)} + n \cdot \ell^2 \cdot 2^{(2\varsigma+1)\ell - \varsigma\ell^2}.$$

Proof. For each choice of $\mathbf{b} \neq \mathbf{b}'$, the probability $\Pr_{h \leftarrow \mathcal{H}}[h(\mathbf{b}) = h(\mathbf{b}')]$ can be expressed in terms of number of zeros of a system of hyperbolic quadratic forms. More precisely let $D = (d_{ij})$ be the $\ell \times \ell$ matrix in $\mathbf{M}_\ell(\mathbb{Z})$ given by $d_{ij} = b_{ij} - b'_{ij}$. We have

$$\begin{aligned}
\Pr_{h \leftarrow \mathcal{H}}[h(\mathbf{b}) = h(\mathbf{b}')] &= \frac{1}{|R|^{2\ell}} \# \left\{ (\mathbf{u}_1, \ldots, \mathbf{u}_\ell, \mathbf{v}_1, \ldots, \mathbf{v}_\ell) \in R^{2\ell} : \sum_{1 \leqslant i,j \leqslant \ell} d_{ij} \cdot \mathbf{u}_i \cdot \mathbf{v}_j = \mathbf{0} \right\} \\
&= \frac{1}{|\mathbb{Z}_{g_1}|^{2\ell}} \# \left\{ (u_{11}, \ldots, u_{\ell 1}, v_{11}, \ldots, v_{\ell 1}) \in \mathbb{Z}_{g_1}^{2\ell} : \sum_{1 \leqslant i,j \leqslant \ell} d_{ij} \cdot u_{i1} \cdot v_{j1} = 0 \right\} \\
&\quad \times \cdots \times \\
&\quad \frac{1}{|\mathbb{Z}_{g_n}|^{2\ell}} \# \left\{ (u_{1n}, \ldots, u_{\ell n}, v_{1n}, \ldots, v_{\ell n}) \in \mathbb{Z}_{g_n}^{2\ell} : \sum_{1 \leqslant i,j \leqslant \ell} d_{ij} \cdot u_{in} \cdot v_{jn} = 0 \right\}
\end{aligned}$$

From [CMNT11, Lemma 4.2], denoting r_i the rank of D in $\mathbb{Z}_{g_i}$, we have

$$\Pr_{h \leftarrow \mathcal{H}}[h(\mathbf{b}) = h(\mathbf{b}')] = \frac{1}{|R|^{2\ell}} \cdot \prod_{i=1}^{n} \left(g_i^{2\ell-1} + g_i^{2\ell-r_i} - g_i^{2\ell-r_i-1} \right).$$

Without loss of generality, we can assume that $g_1 \leqslant g_2 \leqslant \cdots \leqslant g_n$. Thus, we get

$$\Pr_{h \leftarrow \mathcal{H}}[h(\mathbf{b}) = h(\mathbf{b}')] - \frac{1}{|R|} \leqslant \prod_{i=1}^{n} \left(\frac{1}{g_i} + \frac{1}{g_i^{r_i}} \right) - \frac{1}{g_1 \times \cdots \times g_n} \leqslant \frac{1}{g_1^{\min(r_i)} \cdot g_2 \cdots g_n}.$$

As in [CMNT11], this estimate is not sufficient when $\min(r_i) = 1$. Therefore, we need to bound the number of pairs $(\mathbf{b}, \mathbf{b}')$ such that the corresponding matrix D is of rank 1 modulo g_j for at least

one j. Let us denote $U_{\varsigma,j}$ the set of matrices of rank 1 in $M_\ell(\mathbb{Z}_{g_j})$ with entries in $[-2^\varsigma+1, 2^\varsigma-1]$. From [CMNT11], we have the coarse bound $|U_{\varsigma,j}| \leqslant \ell^2 \cdot 2^{(2\varsigma+1)\ell}$ for all j.

Now, the number of pairs $(\mathbf{b}, \mathbf{b}')$ such that the corresponding matrix D is of rank 1 for at least one of the g_j's is at most $n \times |X| \times |U_{\varsigma,j}|$, since for any choice of $\mathbf{b}$ and g_j, there are at most $|U_{\varsigma,j}|$ possible values of $\mathbf{b}'$ such that D is in $U_{\varsigma,j}$. We can thus bound the value δ defined by

$$\delta = \frac{|Y|}{|X|^2} \cdot \sum_{\mathbf{b}\neq\mathbf{b}'} \left(\Pr_{h\leftarrow\mathcal{H}}[h(\mathbf{b}) = h(\mathbf{b}')] - \frac{1}{|R|} \right)$$

as required. Indeed,

$$\begin{aligned}
\delta &\leqslant \frac{|R|}{2^{\varsigma\ell^2}} \cdot \left(\sum_{\substack{\mathbf{b}\neq\mathbf{b}' \\ \forall j, D\notin U_{\varsigma,j}}} \left(\Pr_{h\leftarrow\mathcal{H}}[h(\mathbf{b}) = h(\mathbf{b}')] - \frac{1}{|R|} \right) + \sum_{\substack{\mathbf{b}\neq\mathbf{b}' \\ \exists j, D\in U_{\varsigma,j}}} \left(\Pr_{h\leftarrow\mathcal{H}}[h(\mathbf{b}) = h(\mathbf{b}')] - \frac{1}{|R|} \right) \right) \\
&\leqslant \frac{|R|}{2^{2\varsigma\ell^2}} \cdot \left(2^{2\varsigma\ell^2} \frac{1}{g_1^2 \cdot g_2 \cdots g_n} + n 2^{\varsigma\ell^2} \cdot (\ell^2 \cdot 2^{(2\varsigma+1)\ell}) \cdot \frac{1}{g_1 \cdots g_n} \right) \\
&\leqslant \frac{1}{g_1} + n \cdot \ell^2 \cdot 2^{(2\varsigma+1)\ell - \varsigma\ell^2},
\end{aligned}$$

which concludes the proof. □

Proof of Lemma 12.3. To any encoding $(c \bmod x_0)$ we associate the vector

$$f(c) = \big((c \bmod p_1) \bmod g_1, \ldots, (c \bmod p_n) \bmod g_n\big) \in R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}\,.$$

The function f can be viewed as a decryption function: it extracts from an encoding c the encoded message $\mathbf{m}$. Given two encodings c, c', we have that if $|c \bmod p_i| + |c' \bmod p_i| < p_i/2$ for all i, then $f(c+c') = f(c)+f(c')$. Similarly, if $|c \bmod p_i| \cdot |c' \bmod p_i| < p_i/2$ for all i, then $f(c\cdot c') = f(c)\cdot f(c')$. In the following we only consider encodings which have sufficiently small residues modulo the p_i's so that f can be considered additively and multiplicatively homomorphic.

The quadratic sampling yields an encoding c such that there exists $\mathbf{b}$ and

$$f(c) = f\left(\sum_{1\leqslant i,j\leqslant\ell} b_{ij} \cdot x'_{i,0} \cdot x'_{j,1} \right) = \sum_{1\leqslant i,j\leqslant\ell} b_{ij} \cdot f(x'_{i,0}) \cdot f(x'_{j,1}) = \sum_{1\leqslant i,j\leqslant\ell} b_{ij} \cdot \mathbf{a}_{i,0} \cdot \mathbf{a}_{j,1}\,.$$

Since $\mathbf{a}_{j,b}$ are randomly chosen in R^ℓ, by applying the leftover-hash lemma (Lemma 12.5), we have that $(\mathsf{params}, f(c))$ is ε-statistically close to $(\mathsf{params}, \mathbf{m}')$ for a random $\mathbf{m}'$ in R for

$$\varepsilon = \frac{1}{2}\sqrt{\frac{|R|}{2^{\varsigma\ell^2}} + \epsilon}$$

where ϵ is given in Lemma 12.6. With the condition

$$\varsigma \cdot \ell^2 \geqslant \max\big(n\alpha, (2\varsigma+1)\ell + \log_2(\ell^2) + \log_2(n)\big) + 2\lambda\,,$$

the lemma is proven. □

12.B Optimization on the Zero-Testing Elements

In order to test whether a level-κ encoding encodes $\mathbf{0}$, we publish as part of the instance generation a zero-testing vector $\mathbf{p}_{zt} \in \mathbb{Z}^n$. Unfortunately, this requires to store n integers of $n \cdot \eta$ bits, increasing the public-key size by $n^2 \cdot \eta$ bits. For example with $n = 26115$ and $\eta = 2438$ as in our "Extra" parameters set in Table 12.1, the zero-testing vector size would be larger than 200GB. Moreover, its construction relies on an intricate procedure to generate an invertible matrix $\mathbf{H} \in \mathbb{Z}^{n\times n}$ such that its operator norm, and the operator norm of its inverse are bounded by 2^β (see Appendix 11.B).

In this section, we explain how to reduce the number of elements of $\mathbf{p}_{zt}$ from n to only two and simplify the requirement on $\mathbf{H}$. One drawback is that two encodings of different vectors can now extract to the same value, *i.e.* the scheme is no longer zero-test secure (*cf.* Definition 11.4). While it is actually easy to generate such collisions using LLL, this does not seem to give an attack against the GDDH problem; therefore one-round N-way Diffie-Hellman key exchange as described in Section 12.3 should remain secure under this optimization.

12.B.1 Zero-Testing Element

A first idea would be to publish a zero-testing element p_{zt} as a single integer, instead of a vector $\mathbf{p}_{zt}$ with n components:

$$p_{zt} = \sum_{i=1}^{n} h_i \cdot (z^\kappa \cdot g_i^{-1} \bmod p_i) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0$$

where the h_i's are random integers in $[1, 2^\beta)$. Therefore given as input an integer c such that $c \equiv (r_i \cdot g_i + m_i)/z^\kappa \pmod{p_i}$, we obtain a single integer $\omega = p_{zt} \cdot c \bmod x_0$, with

$$\omega = \sum_{i=1}^{n} h_i \cdot R_i \bmod x_0\,, \tag{12.2}$$

where $R_i = ((r_i + m_i \cdot g_i^{-1}) \bmod p_i) \cdot (x_0/p_i)$. As before, if $\|\mathbf{r}\|_\infty < 2^{\rho_f}$, we still have $|\omega| < x_0 \cdot 2^{-\nu-2}$. However the converse is no longer true: we can have $|\omega| < x_0 \cdot 2^{-\nu+2}$ for an encoding of a non-zero vector $\mathbf{m}$. However if $m_i = 0$ for all $i > 1$, we have the following Lemma, whose proof is similar to the proof of Lemma 11.8:

Lemma 12.7. *Let n, η, α and β be as in our parameter setting. Let ρ_f be such that $\beta + \alpha + \rho_f + \log_2 n \leqslant \eta - 9$, and let $\nu = \eta - \beta - \rho_f - \log_2 n - 3 \geqslant \alpha + 6$. Let c be such that $c \equiv (r_i \cdot g_i + m_i)/z^\kappa \pmod{p_i}$ for all $1 \leqslant i \leqslant n$, where $0 \leqslant m_1 < g_1$ and $m_i = 0$ for all $i > 1$. Let $\mathbf{r} = (r_i)_{1 \leqslant i \leqslant n}$ and assume that $\|\mathbf{r}\|_\infty < 2^{\rho_f}$. If $m_1 = 0$ then $|\omega| < x_0 \cdot 2^{-\nu-2}$. Conversely if $m_1 \neq 0$ then $|\omega| \geqslant x_0 \cdot 2^{-\nu+2}$.*

Proof. If $m_1 = 0$ then we have $R_i = r_i \cdot x_0/p_i$ for all i, which gives using $p_i \geqslant 2^{\eta-1}$ for all i:

$$\|\mathbf{R}\|_\infty \leqslant \|\mathbf{r}\|_\infty \cdot \max_{1 \leqslant i \leqslant n} (x_0/p_i) \leqslant \|\mathbf{r}\|_\infty \cdot x_0 \cdot 2^{-\eta+1}\,.$$

Since by definition $-p/2 < (z \bmod p) \leqslant p/2$, we have $|z \bmod p| \leqslant |z|$ for any z, p; therefore we obtain from (12.2) using $\|\mathbf{r}\|_\infty < 2^{\rho_f}$

$$|\omega| = \|\mathbf{h}^t \cdot \mathbf{R} \bmod x_0\|_\infty \leqslant \|\mathbf{h}^t \cdot \mathbf{R}\|_\infty \leqslant n \cdot \|\mathbf{h}^t\|_\infty \cdot \|\mathbf{R}\|_\infty < x_0 \cdot 2^{\log_2 n + \beta + \rho_f - \eta + 1} = x_0 \cdot 2^{-\nu-2}\,.$$

Conversely assume that $|\omega| < x_0 \cdot 2^{-\nu+2}$. We have from Equation (12.2) that

$$h_1 \cdot R_1 \equiv \omega - \sum_{i=2}^{n} h_i \cdot R_i \pmod{x_0}\,.$$

Now, using the fact that $m_i = 0$ for $i > 1$, we have as previously that $\left|\sum_{i=2}^{n} h_i \cdot R_i\right| \leqslant x_0 \cdot 2^{\log_2(n-1)+\beta+\rho_f-\eta+1} \leqslant x_0 \cdot 2^{-\nu-2}$. Since $|h_1 \cdot R_1| < x_0/2$ and $\left|\omega - \sum_{i=2}^{n} h_i \cdot R_i\right| < x_0 \cdot 2^{-\nu+3}$, the previous equality must hold over $\mathbb{Z}$. This gives $|h_1 \cdot R_1| < x_0 \cdot 2^{-\nu+3}$, and since $h_1 \neq 0$, we get $|R_1| < x_0 \cdot 2^{-\nu+3}$. Letting $v_1 = (r_1 + m_1 \cdot g_1^{-1}) \bmod p_1$, we have $|v_1| \leqslant p_1 \cdot 2^{-\nu+3}$. We show that the equality $g_1 \cdot (v_1 - r_1) \equiv m_1 \pmod{p_1}$ must therefore hold over $\mathbb{Z}$. Indeed from $|m_1| < g_1 < p_1/2$ and $g_1 \leqslant 2^\alpha$, we have

$$|g_1 \cdot (v_1 - r_1)| \leqslant |g_1| \cdot (|v_1| + |r_1|) \leqslant p_1 \cdot 2^{\alpha-\nu+3} + 2^{\alpha+\rho_f} \leqslant p_1/8 + p_1/8 < p_1/2\,,$$

which implies $m_1 \equiv 0 \pmod{p_1}$ and finally $m_1 = 0$. □

Thus if c and c' encode vectors differing only on their first element, by Lemma 12.7 we must have $|(c-c') \cdot p_{zt} \bmod x_0| > x_0 \cdot 2^{-\nu+2}$, and therefore the ν most significant bits of the corresponding ω and ω' must be different. This implies that the min-entropy of $\mathsf{msbs}_\nu(c \cdot p_{zt})$ when c encodes a message $(m_1, m_2, \ldots, m_n)$ for fixed m_i's, $i > 1$ and a random $m_1 \in \mathbb{Z}_{g_1}$ is at least $\log_2|\mathbb{Z}_{g_1}| \geqslant \alpha - 1$. Therefore, the min-entropy of $\mathsf{msbs}_\nu(c \cdot p_{zt})$ when c encodes a random message in R is at least $\alpha - 1$. Finally we can use a strong randomness extractor to extract a nearly-uniform bit-string of length $\alpha - 1 - \lambda$ bits. Thus to extract λ bits, we must have $\alpha \geqslant 2\lambda + 1$, instead of $\alpha = \lambda$ as recommended in Section 11.3.1. The latter bound is therefore not optimal, as a larger α increases the size η of the elements p_i's, and therefore the encoding size.

12.B.2 Extension to $t \leqslant n$ elements

We generalize the previous result to a zero-testing vector with t elements instead of one, namely $\mathbf{p}_{zt} \in \mathbb{Z}^t$ for $1 \leqslant t \leqslant n$:

$$(\mathbf{p}_{zt})_j = \sum_{i=1}^{n} h_{ij} \cdot (z^\kappa \cdot g_i^{-1} \bmod p_i) \cdot \prod_{i' \neq i} p_{i'} \bmod x_0 \,.$$

The matrix $\mathbf{H} = (h_{ij}) \in \mathbb{Z}^{n \times t}$ is randomly generated such that $\mathbf{H} = \begin{pmatrix} \mathbf{H_t} \\ \mathbf{H_{n-t}} \end{pmatrix}$ where the submatrix $\mathbf{H_t} \in \mathbb{Z}^{t \times t}$ is invertible in $\mathbb{Z}$ with both $\|{\mathbf{H_t}}^t\|_\infty \leqslant 2^\beta$ and $\|({\mathbf{H_t}}^{-1})^t\|_\infty \leqslant 2^\beta$ (see Section 11.3 and Appendix 11.B), and the coefficients of $\mathbf{H_{n-t}}$ are random β-bit integers. As previously, if $m_i = 0$ for all $i > t$, we have the following Lemma:

Lemma 12.8. *Let n, t, η, α and β be as in our parameter setting. Let ρ_f be such that $2\beta + \alpha + \rho_f + \log_2(n-t+1) \leqslant \eta - 9$, and let $\nu = \eta - \beta - \rho_f - \log_2(n-t+1) - 3 \geqslant \beta + \alpha + 6$. Let c be such that $c \equiv (r_i \cdot g_i + m_i)/z^\kappa \pmod{p_i}$ for all $1 \leqslant i \leqslant n$, where $0 \leqslant m_i < g_i$ for all $i \leqslant t$ and $m_i = 0$ for all $i > t$. Let $\mathbf{r} = (r_i)_{1 \leqslant i \leqslant n}$ and assume that $\|\mathbf{r}\|_\infty < 2^{\rho_f}$. If $m_i = 0$ for all i then $|\omega| < x_0 \cdot 2^{-\nu-2}$. Conversely if there exists $j \in [1, t]$ such that $m_j \neq 0$ then $|\omega| \geqslant x_0 \cdot 2^{-\nu+2}$.*

Proof. The proof of this Lemma is similar to the proofs of Lemmas 11.8 and 12.7. We sketch it for completeness. If $\mathbf{m} = 0$, we get as previously that $\|\omega\|_\infty \leqslant x_0 \cdot 2^{-\nu-2}$. Assume now that $\|\omega\|_\infty \leqslant x_0 \cdot 2^{-\nu+2}$, and denote $\mathbf{R}^t = ({\mathbf{R_t}}^t, {\mathbf{R_{n-t}}}^t)$. We have that

$$\omega = {\mathbf{H_t}}^t \cdot \mathbf{R_t} + {\mathbf{H_{n-t}}}^t \cdot \mathbf{R_{n-t}} \,.$$

Now $\|\mathbf{H_{n-t}} \cdot \mathbf{R_{n-t}}\|_\infty \leqslant x_0 \cdot 2^{\log_2(n-t)+\beta-\eta+1+\rho_f} \leqslant x_0 \cdot 2^{-\nu-2}$ and

$$\mathbf{R_t} \equiv ({\mathbf{H_t}}^{-1})^t \cdot (\boldsymbol{\omega} - {\mathbf{H_{n-t}}}^t \cdot \mathbf{R_{n-t}}) \bmod x_0 \,,$$

and this latter equation holds over $\mathbb{Z}$. This yields $\|\mathbf{R_t}\|_\infty \leqslant x_0 \cdot 2^{\beta-\nu+3}$ and we conclude as in Lemma 11.8 that $\mathbf{m} = \mathbf{0}$. □

12.B.3 Two-element vector

Let us consider the case $t = 2$. In this case, we do not need to use the intricate generation procedure for $\mathbf{H_2}$ described in Appendix 11.B. Indeed, $\mathbf{H_2} = (h_{ij})$ is invertible over $\mathbb{Z}$ if and only if $h_{11} \cdot h_{22} - h_{12} \cdot h_{21} = \epsilon \in \{\pm 1\}$ and its inverse is given by ${\mathbf{H_2}}^{-1} = \epsilon \begin{pmatrix} h_{22} & -h_{12} \\ -h_{21} & h_{22} \end{pmatrix}$. Therefore if $\|{\mathbf{H_t}}^t\|_\infty \leqslant 2^\beta$, then $\|({\mathbf{H_t}}^{-1})^t\|_\infty \leqslant 2^\beta$.

Now, if c and c' encode vectors $\mathbf{m}$ and $\mathbf{m}'$ with $(m_1, m_2) \neq (m_1', m_2')$ and $m_i = m_i'$ for all $i > 2$, by Lemma 12.8 we must have $\|(c-c') \cdot \mathbf{p}_{zt} \bmod x_0\| > x_0 \cdot 2^{-\nu+2}$, and therefore the ν most significant bits of the corresponding ω and $\boldsymbol{\omega'}$ must be different. This implies that the min-entropy of $\mathsf{msbs}_\nu(c \cdot \mathbf{p}_{zt})$ when c encodes a message $(m_1, m_2, \ldots, m_n)$ for fixed m_i's, $i > 2$ and a random tuple $(m_1, m_2) \in \mathbb{Z}_{g_1} \times \mathbb{Z}_{g_2}$ is at least $\log_2|\mathbb{Z}_{g_1} \times \mathbb{Z}_{g_2}| \geqslant 2(\alpha - 1)$. Therefore, the min-entropy of $\mathsf{msbs}_\nu(c \cdot \mathbf{p}_{zt})$ when c encodes a random message in R is at least $2(\alpha - 1)$. Finally we can use a strong randomness extractor to extract a nearly-uniform bit-string of length $2(\alpha - 1) - \lambda$. Thus to extract λ bits, this implies to take $\alpha \geqslant \lambda + 1$ which is nearly optimal.

Instantiation	λ	κ	n	η	Δ	ρ	$\gamma = n \cdot \eta$	pk size	Setup (once)	Publish (per party)	KeyGen (per party)
Small	52	2	615	897	24	52	$0.5 \cdot 10^6$	14 MB	1.8 s	0.08 s	0.04 s
Medium	62	2	2445	957	49	62	$2.2 \cdot 10^6$	70 MB	20 s	0.45 s	0.22 s
Large	72	2	10080	1027	100	74	$7.3 \cdot 10^6$	330 MB	1008 s	2.4 s	1.2 s
Extra	80	2	14190	1110	119	89	$12.0 \cdot 10^6$	599 MB	3835 s	3.4 s	1.7s

(a) 3-partite Diffie-Hellman

Instantiation	λ	κ	n	η	Δ	ρ	$\gamma = n \cdot \eta$	pk size	Setup (once)	Publish (per party)	KeyGen (per party)
Small	52	4	555	1439	23	52	$0.8 \cdot 10^6$	20 MB	5 s	0.13 s	0.10 s
Medium	62	4	2205	1539	46	62	$3.4 \cdot 10^6$	102 MB	30 s	0.69 s	0.56 s
Large	72	4	8880	1648	94	73	$14.6 \cdot 10^6$	536 MB	1205 s	3.4 s	2.8 s
Extra	80	4	26550	1782	162	87	$47.3 \cdot 10^6$	1.6 GB	24554 s	14.8 s	11.8 s

(b) 5-partite Diffie-Hellman

Instantiation	λ	κ	n	η	Δ	ρ	$\gamma = n \cdot \eta$	pk size	Setup (once)	Publish (per party)	KeyGen (per party)
Small	52	6	525	1981	22	52	$1.0 \cdot 10^6$	26 MB	7 s	0.18 s	0.20 s
Medium	62	6	2055	2121	45	62	$4.4 \cdot 10^6$	133 MB	38 s	0.86 s	1.05 s
Large	72	6	8250	2261	90	72	$18.7 \cdot 10^6$	709 MB	2038 s	4.9 s	5.7 s
Extra	80	6	26115	2438	161	85	$63.7 \cdot 10^6$	2.6 GB	27295 s	17.8 s	20.2 s

(c) 7-partite Diffie-Hellman

Instantiation	λ	κ	n	η	Δ	ρ	$\gamma = n \cdot \eta$	pk size	Setup (once)	Publish (per party)	KeyGen (per party)
Small	52	25	405	7130	20	52	$2.9 \cdot 10^6$	72 MB	9.5 s	0.52 s	2.2 s
Medium	62	25	1590	7650	39	62	$12.1 \cdot 10^6$	361 MB	190 s	2.9 s	12.1 s
Large	72	25	6285	8170	79	72	$51.3 \cdot 10^6$	2.0 GB	5321 s	16.9 s	74.4 s
Extra	80	25	18990	8637	137	81	$164.0 \cdot 10^6$	8.3 GB	123633 s	59.1 s	254.5 s

(d) 26-partite Diffie-Hellman

Table 12.1 – Parameters and timings to instantiate a *one-round N-way Diffie-Hellman key exchange protocol* with $\ell = 160$, $\beta = 80$, $\alpha = 80$, $N = \kappa + 1$ and $\nu = 160$ on a 16-core computer (Intel(R) Xeon(R) CPU E7-8837 at 2.67GHz) using GMP 6.0.0. We denote by γ the bitsize of the encodings. Note that the Setup step was parallelized on the 16 cores to speed-up the process while the other steps ran on a single core. We only derived a common ν-bit session key without using a randomness extractor.

Part Four

Conclusions, Thoughts and Other Works

Chapter 13

Conclusion and Thoughts

The huge gap that exists between the scientific state of the art in cryptography and the cryptographic systems embedded in current security products never fails to astonish me.

One example, among numerous others, concerns the *Transport Layer Security* (TLS) protocol. In February 2013, the TLS world was mostly running on RC4-SHA (48.9%) and AES-CBC (47.5%) [Lanb].[1] During the year were presented the Lucky 13 attack [AP13] against all TLS and DTLS ciphersuites that include CBC-mode encryption, and an attack using RC4 biases in TLS [ABP+13]. These attacks are quite efficient and make clear that both these major TLS ciphersuite families, *which constitute of 96.4% of the Internet*, shall be replaced as soon as possible. A possible countermeasure (for both attacks) consists in switching to AEAD ciphersuites[2] in TLS, such as AES-GCM. Support for AEAD ciphersuites was specified in TLS 1.2, but this version of TLS was not, *and is still not*, widely supported. Moreover, producing a fast AES-GCM implementation which is not prone to timings attacks is very difficult, and AES-GCM is quite unadapted to low-powered devices [Lana]. And if that were not enough, the RC4-SHA and AES-CBC ciphersuites cannot be deactivated because of a long-standing problem: *fallback* mechanisms. Fallback modes allow, when a handshake fails (for example because of a – maliciously? – buggy HTTPS server), to reconnect with a less secure version of the protocol.

This example is *one of many* situations in which *"old"* cryptography is used instead of more recent primitives with strong security guarantees. Only a small fraction of recent research in cryptography is really used in practice; the majority of the cryptographic protocols lack implementations and performance measures (and often lack help on parameters selection). Advanced cryptography can do much more for security applications than just using AES, RSA and ECC – and I believe one of the great challenge of cryptology in the following years is to find a way to bring recent advances in real world products in a reasonable time frame, and to find solutions to ensure security when weak primitives are used[3].

I had been very lucky to be offered the opportunity of doing most of my Ph.D. studies at CryptoExperts, a small company that intends to narrow – and fill – the gap between theoretical research and practical cryptography. I am also admiring the works of Adam Langley[4], Daniel J. Bernstein and Tanja Lange[5] who aim at making the world more secure. Recent cryptographic competitions such as the SHA-3 and CAESAR competitions [NIS12, CAE16] aim at designing

[1] RC4 is a secret stream cipher designed *in 1987* (and leaked in 1994 [Ano94]) and has become part of some commonly used encryption protocols and standards. Its wide adoption is mainly due to its speed and simplicity; but not to its security, as RC4 is known to have a variety of cryptographic weaknesses [GMPS14].

[2] Authenticated Encryption with Associated Data (AEAD) is a class of block cipher modes which encrypt (parts of) the message and authenticate the message simultaneously.

[3] A very interesting work on this subject, that received the *best young-author paper* at CRYPTO 2013, is the *counter-cryptanalysis* paradigm proposed by Marc Stevens [Ste13]. Counter-cryptanalysis "exploits unavoidable anomalies introduced by cryptanalytic attacks to detect and block cryptanalytic attacks while maintaining full backwards compatibility".

[4] Adam Langley works at Google, and does an amazing job at bringing recent cryptographic primitives (such as ChaCha20 and Poly1305) into Chrome and on Google servers [Lana, Lanb]; his work impacts millions of people.

[5] Dan and Tanja are very active and passionate members of the cryptographic community. Among other works, they are doing an amazing job assessing the security of implementing elliptic-curves in cryptography [BL].

trusted cryptography for standardization that is fast, efficient, and resists to the latest advances in cryptanalysis. Another dissemination effort that will certainly have a huge impact on this issue is the recent Real World Cryptography workshop [RWC] which focuses on uses of cryptography in real world environments, and brought together more than 400 participants for its 2014 edition.

Throughout my Ph.D. studies, my main research inclination was inspired by bridging the gap between the theoretical and practical worlds. My work aims to help some of the most recent cryptographic primitives (lattice-based cryptography, fully homomorphic encryption and multilinear maps schemes) become more practical. I endeavor, in the *long run*, to make these supposedly *utterly ludicrously slow* primitives (as emphasized *e.g.* in [Ber]) useful in real world applications.

Thoughts on Lattice-Based Cryptography. Lattice-based cryptography is not a recent field. Quite surprisingly[6], the most promising candidate for lattice-based encryption remains NTRUEncrypt, presented in 1998 [HPS98]. NTRUEncrypt is standardized [IEE08] and has resisted cryptanalytic attention for the last 16 years. However, repeated attacks on lattice-based signatures casted doubts on the maturity of lattice-based cryptography in the community. One of the main problems was certainly that these cryptographic primitives lacked security proofs, and therefore lacked evidence of their hardness. However, in the last 10 years, the landscape changed considerably, starting from the work of Regev [Reg09] that introduces the LWE average-case problem and shows that it reduces to worst-case algorithmic problems over lattices. Note that algorithms over Euclidean lattices has been extensively studied independently of any cryptographic applications because numerous optimization problems rely on it. The known hardness of these problems, and the worst-case to average-case reduction, prove that almost all the lattices we will rely on are equally hard, contrary to elliptic curves or finite field cryptographies in which some parameters choices can end-up yielding a system a lot easier than expected [BGJT14]. Following Regev's groundbreaking work, lattice-based cryptography fully entered modern cryptography by providing strong arguments (and often rigorous proofs) of its security.

Lattice-based cryptography is *amazingly versatile*, almost all known – if not all? – cryptographic primitives can be instantiated from lattices. Unfortunately, this versatility comes at the cost of simplicity; it seems really difficult to use lattice primitives as *black boxes*. For example, in his Ph.D. thesis [Duc13], Ducas discussed this fact when instantiating an HIBE (Hierarchical Identity-Based Encryption) scheme. Such a scheme can be instantiated from *one* group using pairings, independently of the hierarchical level h, while the lattice-based instantiation requires different parameters according to the value of h (with contributions often hidden in the Landau notation $\tilde{\mathcal{O}}$, extensively used by theoretical cryptographers). This fact is also illustrated throughout Part I of this thesis. Opening the "black-box" allowed to propose a lot of optimizations that are specific to our Fiat-Shamir based signature scheme (bimodal Gaussians, compression, use of non-uniform lattices similar to NTRU lattices). Lattice-based cryptographic schemes require *numerous* parameters that are intricately linked together, and that affect tremendously the complexity of the known attacks and the efficiency of the schemes. Selecting parameters ensuring both correctness and a given level of security is a delicate optimization problem, that we tackled for the selection of the parameters of BLISS in Part I (*cf.* Chapter 6) and in [LN14a]. To optimize our parameters for BLISS (and throughout all the thesis), we used the mathematical software `SAGE` [S$^+$14]. I believe that automatic optimization to select parameters will be unavoidable in the future because of the numerous constraints due to correctness (of the zillion of lattice-based schemes) and to security.

Lattice-based cryptography is repeatedly claimed throughout the literature to be computationally simple – as it relies on very simple operations (contrary to elliptic curves operations or exponentiation modulo a large number) –, powerful and very efficient. Unfortunately, this simplicity is largely oversold in my opinion, and one lesson I learned from this thesis is that making the mathematically elegant lattice-based schemes practical is an *intricate* task.[7] One of the first issue concerns one of the building block of lattice-based cryptography, the discrete Gaussian sampling. The general

[6] Vadim Lyubashevsky emphasized in a series of talks about lattice-based encryption [Lyu] that the timeline of lattice-based cryptography is illogical: NTRUEncrypt was discovered first in 1998, although it follows naturally from a series of works [Reg09, LPR13a, SS11b].

[7] In light of this information, the lack of parameter choices and concrete instantiations from the lattice-based cryptographic community makes more sense (but is still irksome).

discrete Gaussian sampling still requires to work with floating-point numbers [DN12a], which gainsay the simple operations (matrix/vector operations) of lattice-based cryptography. Fortunately our signature scheme BLISS (described in Part I of this thesis) only requires discrete Gaussian sampling over the integers. In Chapter 4, we proposed simple algorithms (adapted to constrained devices) to realize this sampling with small memory footprint and nearly optimal entropy consumption. Another daunting issue is that there are actually two types of lattice-based cryptography: lattices are either random or have a strong mathematical structure (*i.e.* are ideal lattices). Unfortunately, working with random lattices makes schemes *much* slower and public keys *much* larger, and is therefore sadly incompatible with efficiency. Therefore this begs the question of the hardness of algorithmic problems over *ideal* lattices. Currently, it is not known how to significantly exploit the additional structure in a lattice compared to a random lattice (this might dramatically change in the future [Ber]) and parameter selection relies on the best attacks on random lattices currently known. Moreover a lot of cryptanalytic effort is dedicated to general lattice reduction, and to algorithmic problems over lattices (such as SVP) and not often to the lattice-based cryptosystems themselves. My belief is that this lack of cryptanalytic effort is in part due to the fact that most of the papers that propose new schemes give no concrete targets to attack and only provide asymptotic parameters constraints. My hope is that all the parameters suggested for BLISS in Chapter 6 (and those for the schemes in Parts II and III, that can be attacked using lattice reduction algorithms) make it "worthwhile" for cryptanalysts to work over these problems. In short, one of my hopes is that the results described in this thesis spur the cryptanalysis that is currently *much* needed in the field.

Despite the abovementioned concerns, I am really enthusiastic about the potential of lattice-based cryptography. Some recent implementation efforts (I would like to mention particularly the works of Thomas Pöpplemann [GLP12, PG12, PG13, PG14, OPG14] and his co-authors) bring a lot of confidence in the efficiency of classical primitives such as encryption and signature even for small architectures. A recent work suggests that the BLISS signature scheme (*cf.* Chapter 6) is more efficient *on every aspect* compared to ECDSA and RSA on reconfigurable hardware (with a signature size of 5600 bits) [PDG14]. Even though a lot of work remains to construct secure "real world" lattice-based schemes (therefore also resistant to side-channel attacks) – because of the novelty of these algorithms compared to more classical cryptography –, I believe striking results will be obtained in the next decade.

Thoughts on Fully Homomorphic Encryption. The landscape for fully homomorphic encryption has undoubtedly changed in the last five years. In 2009, Gentry theoretically described the first FHE scheme [Gen09] over ideal lattices that relies on *very* strong security assumptions. The first implementation of this scheme is described in 2011 by Gentry and Halevi [GH11b] and each bit-multiplication requires a 30-minute refreshing procedure before any subsequent operation is possible. In 2014, there exist at least four big families of FHE schemes, three of them backed with more or less efficient implementations, and full-fledged homomorphic evaluations of lightweight block ciphers (namely Simon and Prince) run in a matter of minutes on mid-range computers. Although these speed records are rather slow (and might even be considered as *ludicrously slow* [Ber]), homomorphic evaluations of *shallow* circuits (*i.e.* of constant depth and polynomial size) are becoming really efficient [NLV11, BLLN13, LN14a]. These latter circuits allow for example to perform statistic computations (such as the mean[8], the variance or statistical tests such as linear regressions [NLV11]) over encrypted data, or machine learning algorithms [GLN12]. In particular, using homomorphic encryption could prove really useful for *numerous* real world applications on medical data, biometrics or localization as long as the multiplicative depth of the circuit remains small.[9]

Homomorphic encryption is also considered as the main element to secure the "cloud". Unfortunately the huge ciphertext expansion (*i.e.* the size of the ciphertext compared to the size of the plaintext) of current schemes makes it prohibitive to send all of a client data in the cloud encrypted under an FHE scheme. Hybrid solutions, in which data are sent encrypted under an encryption scheme with no ciphertext expansion (*e.g.* AES [GHS12c, CCK+13], Simon [LN14a],

[8] In these statistics computations, one computes independently the numerator and the denominator to avoid the unnecessary complicated division. Note that since the mean can be computed only from homomorphic additions, a simply homomorphic encryption scheme such as the Paillier scheme [Pai99] would suffice.

[9] To give an order of magnitude, let us give the rough estimate of a multiplicative depth of 30 for today's schemes.

Prince [DSES14]) and then homomorphically decrypted before being processed, is a current mainstream subject (in which I have contributed for AES and Simon). Unfortunately, as mentioned in Section 10.4.1 (page 135) and in a common work with Michael Naehrig [LN14a], dividing the total time by the number of blocks processed in parallel in a single homomorphic evaluation might not be really meaningful to assess the practicality of homomorphic encryption for real world uses. One should rather focus on optimizing the latency (*i.e.* the time required to perform the entire homomorphic evaluation), as the throughput can be increased as much as needed using parallel computing. Note that it is not clear currently what is the best solution with small latency. Using Prince as a block cipher is a really exciting possibility as it has a multiplicative depth of "only" 24 levels, but new works in progress suggest that using variants of the one-time pad and a pseudo-random key stream might be far more efficient for real world uses. However, achieving *fully* homomorphic encryption currently remains prohibitive in practice, as using bootstrapping is required in all existing FHE schemes.

Homomorphic encryption can no longer be considered fully impractical as for today. The European Commission explicitly mentioned the construction of "Resource efficient, real-time, highly secure fully homomorphic cryptography" as a key challenge in its last Information and Communications Technologies call for projects [H20]. This industrially oriented call proves that homomorphic encryption is considered as a really promising key feature for the *near future* and should be investigated for practical deployment.

Thoughts on Multilinear Maps. Cryptographic multilinear maps, a generalization of the very fruitful bilinear maps [Jou00, SOK00, BF01], were considered back in 2003 by Boneh and Silverberg [BS03]. Even nowadays, constructing such a scheme remains an open problem. However, in a breakthrough work in 2013, Garg, Gentry and Halevi described a new primitive, called *graded encoding systems*, that can be viewed as a generalization of bilinear maps that differs from Boneh and Silverberg generalization. Their candidate multilinear maps scheme still allows to do a N-multipartite Diffie-Hellman key exchange for any number N of users, and loads of new applications based on this primitive were proposed these last months. Chiefly among them is certainly the candidate indistinguishability obfuscation primitive iO of Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH$^+$13b]. Roughly speaking, the definition of indistinguishability obfuscation states that given any two equivalent circuits C_0 and C_1 of "similar size" (with exactly the same input/output behavior), the obfuscations of C_0 and C_1 should be computationally indistinguishable. Indistinguishability obfuscation for circuits, public-key encryption, and non-interactive zero knowledge also allowed the authors to achieve functional encryption for all circuits, which was likewise a long standing open problem. I am also interested in looking into the relation between white-box cryptography and indistinguishability obfuscation, *cf.* Appendix A.

A cryptographic multilinear map is a very recent primitive with a tremendous potential, but it faces many open problems that cannot be ignored. In particular, both the multilinear maps candidates of [GGH13a] and described in Part III of this thesis are not backed up with security proofs (which is unusual in modern public key cryptography). Essentially, the hardness assumption of these schemes is... that the schemes are secure! Both works present cryptanalytic arguments, based on state of the art in cryptanalysis, to justify the current hardness of the schemes. It would be a major result to construct a multilinear maps scheme based on standard assumptions (such as LWE).[10] Another ineludible problem is the practicality of these schemes. With Jean-Sébastien Coron and Mehdi Tibouchi, we proposed the *first* proof-of-concept implementation of multilinear maps [Lep13] as described in Chapter 12, but even our heuristic optimizations only allowed us to consider a 6-linear map while ensuring arguably reasonable timings (40 seconds for a one-round key exchange between 7 parties). However, due to the similarity of multilinear maps with fully homomorphic encryption, I am confident that we have a lot of room for improvement (as illustrated by the current FHE schemes timings on shallow circuits). Also, I wonder whether

[10] On a side – but related – note, both multilinear maps candidates can be seen as extensions of SHE schemes. Namely, the scheme of Garg *et al.* is similar to Gentry's FHE scheme (mixed with NTRU-like constructions), while the scheme of Part III is really similar to the multi-slot DGHV scheme proposed in Chapter 7. One of the main problems to provide a security proof is the additional zero-testing vector which leaks some information *by design*. It remains an open problem – and I plan to look into it in the near future – to obtain multilinear maps from the two other main SHE schemes [BV11a] and [GSW13].

the use of FHE tricks, such as modulus-switching or scale-invariance, would improve the efficiency and would allow to obtain encodings of constant size (or logarithmic) instead of polynomial in the multilinearity level. I predict that in 5 years, multilinear maps with 20 to 30 levels will run in a few milliseconds on mid-range computers (instead of seconds to minutes with the current implementation, *cf.* Chapter 12). However, even these practicality improvements would be far behind what a lot of applications require (even without considering indistinguishability obfuscation that requires thousands of thousands of levels). I suspect that applications with a small number of levels might become fully practical in the near future, and I therefore spur the community to work on applications with a small number of multilinearity levels.

About Noisy Cryptography. Last but not least, I would like to share my thoughts about "noisy" cryptography. Fully Homomorphic Encryption and Multilinear Maps schemes rely on a design principle, in which ciphertexts (resp. encodings – that can be viewed as encryptions of scalar values) contain some noise which grows with successive operations. The downsides of this principle are *multiple*; in the following I will briefly emphasize two of them. First, the exponential noise growth in the first generation of FHE schemes and both multilinear maps candidates *severely* limits the number of possible operations in order to keep reasonable timings. (And even the linear growth for the second generation of FHE scheme does not allow to consider too many levels.) And secondly, independently of this practical obstacle, this design paradigm implies that the parameters of the schemes *depend on the underlying protocol* and cannot be specified once and for all (contrary to RSA or elliptic curves).

As a consequence parameter selection is a *critical* and *intricate* task. I do believe the cryptographic community needs to build *models* that would allow to automatically choose parameters according to the protocol and desired security level. To my knowledge, no work except the work presented in Chapter 9, where we propose an algorithm to determine how to bootstrap based on a simplified model of the noise growth in FHE schemes, is tackling this issue.

Final Thoughts. Cryptography is a young, fast moving and critical field. Unfortunately, the profusion of exciting theoretical results has currently limited impact on "real world" cryptography. Fortunately, more and more people aim at breaking the wall between theory and practice. This goes from high-speed implementations of modern primitives on numerous architectures to contributions to standardization. I hope my work on mitigating theory and practice did contribute to fill the gap between these two aspects of cryptography, and I intend to continue working on bridging the exciting theoretical results with high-speed secure implementations. My main objective is to ensure privacy while providing a rich and seamless experience to the end-users.

Appendix A

Other Works on White-Box Cryptography

White-box cryptography was introduced in 2002 by Chow, Eisen, Johnson and van Oorschot as the ultimate, *worst-case* attack model [CEJvO02b, CEJvO02a]. This model considers an attacker far more powerful than in the classical black-box model (and thus more representative of real-world attackers); namely the attacker is given *full knowledge* and *full control* on both the algorithm and its execution environment. However, even such powerful capabilities should not allow her to *e.g.* extract the embedded key.[1] White-box cryptography can hence be seen as a restriction of general obfuscation where the function to protect belongs to some narrower class of cryptographic functions indexed by a secret key. From that angle, the ultimate goal of a white-box implementation is to leak nothing more than what a black-box access to the function would reveal. An implementation achieving this strong property would be as secure as in the black-box model, in particular it would resist *all existing and future* side-channel and fault-based attacks. Although we know that general obfuscation of any function is impossible to achieve [BGI+01], there is no known impossibility result for white-box cryptography and positive examples have even been discovered [HRSV07, CCV12]. The work of Chow and others gave rise to several proposals for white-box implementations of symmetric ciphers, specifically DES [CEJvO02b, LN05, WP05] and AES [CEJvO02a, BCD06, XL09, Kar10], even though all these proposals have been broken [JBF02, BGEC04, GMQ07, WMGP07, MGH08, MWP10, MRP12, LRM+13].

In [CEJvO02b, CEJvO02a], Chow *et al.* proposed a generic strategy to produce white-box implementations of (symmetric) cryptographic algorithms, and for which key extraction was supposedly hard. From this strategy, they derive two candidate white-box implementations of DES and AES. The followed approach is that look-up tables might be the ideal primitives to hide information, since they allow to implement any given function. They proposed to express the algorithm into a network of look-up tables which, combined all together, yield the complete cryptographic algorithm. To avoid information leakage (and thus key leakage), the basic idea is to compose each table, on input and on output, with random bijections that annihilate one with another when composed.

Our Works. In [LRM+13], we show that the *last* (in 2013) candidate white-box AES implementation due to Karroumi [Kar10] can be broken by a direct application of Billet *et al.* attack [BGEC04].[2] We also describe an improved version of the latter attack and a new, conceptually simpler attack, both of complexity 2^{22}. Our new cryptanalysis technique exploits collisions (or zero-values) in output of the first round in order to construct sparse linear systems. Solving these systems then reveals the input encoding and secret key byte(s) involved in some target look-up table. Applied to the original scheme, we get an attack of complexity 2^{22}, conceptually simpler than all previous attacks [BGEC04, MGH08].

Although all practical white-box candidates have been broken, neither evidence of existence nor proofs of impossibility have been provided for this particular setting. This might be in part because

[1] Quoting [CEJvO02b], the "choice of the implementation is the sole remaining line of defense and is precisely what is pursued in white-box cryptography".

[2] Specifically, we show that for any given secret key, the overall implementation has the *exact same* distribution as the implementation of Chow *et al.* making them both vulnerable to the same attacks.

it is still quite unclear what white-box cryptography really aims to achieve and which security properties are expected from white-box programs in applications. Therefore in [DLPR13b] we build a first step towards a practical answer to this question by translating folklore intuitions behind white-box cryptography into concrete security notions. Specifically, we introduce the notion of white-box compiler that turns a symmetric encryption scheme into randomized white-box programs, and we capture several desired security properties such as one-wayness, incompressibility and traceability for white-box programs. We also give concrete examples of white-box compilers (coming from the realm of public-key cryptography) that already achieve some of these notions.[3] Overall, our results open new perspectives on the design of white-box programs that securely implement symmetric encryption.

Conclusion and Thoughts. Recently, a candidate construction of *indistinguishability obfuscation iO* has been proposed by Garg, Gentry and Halevi [GGH13a]. In particular, using *iO* on two circuits with the same input/output behavior and of the "same size", it is unfeasible for an adversary to distinguish the input circuit. It is worth noting that *iO* might allow to understand white-box cryptography better. In particular, assume there exists a white-box implementation of AES which is unbreakable (*i.e.* it is unfeasible to recover the embedded AES key). If this implementation C_0 has the same size as a classical AES implementation C_1 with the same key, then it should be unfeasible to distinguish the obfuscations of C_0 and C_1 when using the *iO* primitive. In particular, it should be unfeasible from the obfuscation of C_1 (*i.e.* the obfuscation of the *classical* AES implementation) to recover information on the key because of the unbreakability assumption on C_0. In a way, indistinguishability obfuscation is the "best possible obfuscation" [GR07] and might be an interesting candidate for white-box cryptography.

In practice, obfuscating AES using *iO* is currently *completely prohibitive*: in [Cor13], Coron estimated an obfuscated AES evaluation to take $2 \cdot 10^{62}$ years. On the other hand, our last attacks [LRM+13] cast doubt on the approach to design a (white-box) obfuscation of AES by a network of lookup tables. It remains therefore a challenging problem to design a new approach to obfuscate AES, that is efficient and so that the implementation is at least unbreakable (*i.e.* one cannot recover the embedded secret key). However, I think that the similarities between Kilian's matrix product randomization technique [Kil88] (used in *iO*) and the randomization of the network of lookup tables are mind-shattering. In particular, it could be worth investigating whether the additional randomizations techniques used in [GGH+13b] to achieve *iO* could be transposed the look-up tables network approach.

White-box cryptography is definitely a very compelling subject, as an efficient construction would resist to any present *and future* side-channel attack. We hope that our works on the subject will make it "worthwhile" for cryptographers to work on this exciting issue.

[3] For the first two notions, we show an example of a simple symmetric encryption scheme over an RSA group for which an efficient white-box compiler exists that provably achieves both notions. We finally show that white-box programs are efficiently traceable by simple means assuming that functional perturbations can be hidden in them.

List of Figures

List of Tables

List of Algorithms

Bibliography

[Abe10] Masayuki Abe, editor. *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*. Springer, 2010.
→ Cited on pages 204 and 205.

[ABP+13] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C.N. Schuldt. On the security of RC4 in TLS and WPA. In *USENIX Security Symposium*, 2013.
→ Cited on page 173.

[ABS13] Andrew A. Adams, Michael Brenner, and Matthew Smith, editors. *Financial Cryptography and Data Security - FC 2013 Workshops, USEC and WAHC 2013, Okinawa, Japan, April 1, 2013, Revised Selected Papers*, volume 7862 of *Lecture Notes in Computer Science*. Springer, 2013.
→ Cited on pages 17, 119, 198, and 200.

[AGHS13] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Discrete Gaussian leftover hash lemma over infinite domains. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (1)*, volume 8269 of *Lecture Notes in Computer Science*, pages 97–116. Springer, 2013.
→ Cited on pages 142 and 152.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In Canetti and Garay [CG13b], pages 500–518.
→ Cited on page 15.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Miller [Mil96], pages 99–108.
→ Cited on pages 2, 15, and 25.

[Alb14] Martin Albrecht. Cryptanalysis of the FHE based on GACD?, 2014. `http://martinralbrecht.wordpress.com/2014/02/18/cryptanalysis-of-the-fhe-based-on-gacd/`, accessed 12 May 2014.
→ Cited on page 90.

[AMW07] Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors. *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, volume 4876 of *Lecture Notes in Computer Science*. Springer, 2007.
→ Cited on pages 194 and 206.

[Ano94] Anonymous. Source code of RC4, 1994. `http://web.archive.org/web/20080404222417/http://cypherpunks.venona.com/date/1994/09/msg00304.`

html, accessed 3 June 2014.
→ Cited on page 173.

[AP13] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *IEEE Symposium on Security and Privacy*, pages 526–540. IEEE Computer Society, 2013.
→ Cited on page 173.

[ASP13] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In Canetti and Garay [CG13a], pages 1–20.
→ Cited on page 73.

[ASP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. *IACR Cryptology ePrint Archive*, 2014:94, 2014. To appear at CRYPTO 2014.
→ Cited on page 77.

[Bab64] Charles Babbage. *Passages from the life of a philosopher*. Longman, Green, Longman, Roberts, & Green, 1864.
→ Cited on page 11.

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.
→ Cited on page 157.

[BB13] Rachid El Bansarkhani and Johannes Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In Lange et al. [LLL13].
→ Cited on pages 29, 40, 41, and 71.

[BCD06] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. White box cryptography: Another attempt. *IACR Cryptology ePrint Archive*, 2006:468, 2006.
→ Cited on pages 8 and 179.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. Prince - a low-latency block cipher for pervasive computing applications - extended abstract. In Wang and Sako [WS12], pages 208–225.
→ Cited on pages 8 and 137.

[BCG+13] Johannes Buchmann, Daniel Cabarcas, Florian Göpfert, Andreas Hülsing, and Patrick Weiden. Discrete Ziggurat: A time-memory trade-off for sampling from a Gaussian distribution over the integers. Cryptology ePrint Archive, Report 2013/510, 2013. http://eprint.iacr.org/.
→ Cited on page 40.

[BDHG99] Dan Boneh, Glenn Durfee, and Nick Howgrave-Graham. Factoring $N = p^r q$ for large r. In Wiener [Wie99], pages 326–337.
→ Cited on page 81.

[Ben64] Václad E. Beneš. Optimal rearrangeable multistage connecting networks. *Bell Systems Technical Journal*, 43(7):1641–1656, 1964.
→ Cited on page 101.

[Ber] Daniel J. Bernstein. A subfield-logarithm attack against ideal lattices. http://blog.cr.yp.to/20140213-ideal.html, accessed 3 June 2014.
→ Cited on pages 4, 6, 174, and 175.

[Ber08] Daniel J. Bernstein. Fast multiplication and its applications. In Joe Buhler and Peter Stevenhagen, editors, *Algorithmic number theory: lattices, number fields, curves and cryptography*, pages 325–384. Cambridge University Press, 2008.
→ Cited on page 58.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Kilian [Kil01], pages 213–229.
→ Cited on pages 6, 15, 139, and 176.

[BG14] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *CT-RSA*, volume 8366 of *Lecture Notes in Computer Science*, pages 28–47. Springer, 2014.
→ Cited on page 72.

[BGEC04] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a white box AES implementation. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2004.
→ Cited on pages 8 and 179.

[BGH13] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in LWE-based homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2013.
→ Cited on pages 4, 73, 77, 79, and 101.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Kilian [Kil01], pages 1–18.
→ Cited on page 179.

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In Nguyen and Oswald [NO14], pages 1–16.
→ Cited on page 174.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.
→ Cited on page 75.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS*, pages 309–325. ACM, 2012.
→ Cited on pages 1, 3, 4, 5, 26, 73, 77, 80, 107, 110, 119, 120, 123, 129, and 135.

[BHL13] Daniel J. Bernstein, Nadia Heninger, and Tanja Lange. The year in crypto, 2013. 30th Chaos Communication Congress.
→ Cited on page 73.

[Bih97] Eli Biham. A fast new DES implementation in software. In Eli Biham, editor, *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 1997.
→ Cited on pages 129, 130, and 133.

[BL] Daniel J. Bernstein and Tanja Lange. SafeCurves: choosing safe curves for elliptic-curve cryptography. `http://safecurves.cr.yp.to`, accessed 3 June 2014.
→ Cited on page 173.

[BLLN13] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Stam [Sta13], pages 45–64.
→ Cited on pages 6, 8, 32, 73, 77, 78, 119, 123, 137, and 175.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Boneh et al. [BRF13], pages 575–584.
→ Cited on pages 15, 25, 26, and 31.

[BMP13] Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *J. Cryptology*, 26(2):280–312, 2013.
→ Cited on pages 119, 121, 125, 126, 133, and 134.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 390–399. ACM, 2006.
→ Cited on pages 45 and 50.

[BP02] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2002.
→ Cited on pages 2 and 42.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES S-Box. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *SEC*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer, 2012.
→ Cited on pages 121, 125, and 126.

[BP13] Fabrice Benhamouda and David Pointcheval. Verifier-based password-authenticated key exchange: New models and constructions. *IACR Cryptology ePrint Archive*, 2013:833, 2013.
→ Cited on pages 2, 140, 142, and 156.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Safavi-Naini and Canetti [SNC12], pages 868–886.
→ Cited on pages 5, 32, 73, 74, 77, 107, 108, 117, and 123.

[BRF13] Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors. *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. ACM, 2013.
→ Cited on pages 187 and 194.

[BS03] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.
→ Cited on pages 6, 12, 13, 139, 141, 143, 159, 161, and 176.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptology ePrint Archive*, 2013:404, 2013.
→ Cited on pages 8 and 137.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Ostrovsky [Ost11], pages 97–106.
→ Cited on pages 32, 73, 76, 77, 101, 119, 121, 123, and 176.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Rogaway [Rog11], pages 505–524.
→ Cited on pages 5, 32, 73, 77, 78, and 101.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS*, pages 1–12. ACM, 2014.
→ Cited on pages 73 and 77.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Sako and Sarkar [SS13], pages 280–300.
→ Cited on page 139.

[BZ13] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2013:642, 2013. To appear at CRYPTO 2014.
→ Cited on page 139.

[CAE16] CAESAR: Competition for authenticated encryption: Security, applicability, and robustness, 2013-2016. http://competitions.cr.yp.to/caesar.html.
→ Cited on pages 11 and 173.

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Johansson and Nguyen [JN13], pages 315–335.
→ Cited on pages v, 4, 5, 8, 9, 18, 73, 75, 80, 81, 86, 119, 120, 121, 123, 129, 142, and 175.

[CCV12] Nishanth Chandran, Melissa Chase, and Vinod Vaikuntanathan. Functional re-encryption and collusion-resistant obfuscation. In Ronald Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 404–421. Springer, 2012.
→ Cited on page 179.

[CEJvO02a] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-box cryptography and an AES implementation. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270. Springer, 2002.
→ Cited on pages 8 and 179.

[CEJvO02b] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A white-box DES implementation for DRM applications. In Feigenbaum [Fei03], pages 1–15.
→ Cited on pages 8 and 179.

[CG13a] Ran Canetti and Juan A. Garay, editors. *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*. Springer, 2013.
→ Cited on pages 17, 18, 31, 41, 53, 141, 159, 186, 190, 191, 192, 194, 196, 200, and 205.

[CG13b] Ran Canetti and Juan A. Garay, editors. *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*. Springer, 2013.
→ Cited on pages 185 and 193.

[CGPV10] Jean-Sébastien Coron, Aline Gouget, Pascal Paillier, and Karine Villegas. SPAKE: A single-party public-key authenticated key exchange protocol for contact-less applications. In Radu Sion, Reza Curtmola, Sven Dietrich, Aggelos Kiayias, Josep M. Miret, Kazue Sako, and Francesc Sebé, editors, *Financial Cryptography Workshops*, volume 6054 of *Lecture Notes in Computer Science*, pages 107–122. Springer, 2010.
→ Cited on page 81.

[CH12] Henry Cohn and Nadia Heninger. Approximate common divisors via lattices. In *ANTS X*, 2012.
→ Cited on pages 81, 84, and 87.

[cKK09] Çetin Kaya Koç, editor. *Cryptographic Engineering*. Springer, 2009.
→ Cited on pages 196 and 203.

[CLT13a] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Batch fully homomorphic encryption over the integers. *IACR Cryptology ePrint Archive*, 2013:36, 2013.
→ Cited on pages 9, 18, 75, 98, 129, and 142.

[CLT13b] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Canetti and Garay [CG13a], pages 476–493.
→ Cited on pages v, 6, 7, 9, 18, 86, 98, 141, 159, and 164.

[CLT13c] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. *IACR Cryptology ePrint Archive*, 2013:183, 2013.
→ Cited on pages 9, 18, 141, and 159.

[CLT14a] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In Krawczyk [Kra14], pages 311–328.
→ Cited on pages v, 4, 5, 8, 9, 18, 73, 75, 80, 107, 109, 123, 129, and 142.

[CLT14b] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. *IACR Cryptology ePrint Archive*, 2014:032, 2014.
→ Cited on pages 9, 18, 73, 107, 109, and 129.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Rogaway [Rog11], pages 487–504.
→ Cited on pages 1, 4, 73, 76, 77, 78, 80, 81, 84, 85, 86, 87, 89, 90, 91, 92, 93, 94, 96, 98, 99, 100, 104, 106, 108, 112, 119, 120, 121, 123, 160, 162, 163, 164, 165, and 166.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.
→ Cited on pages 23, 24, 52, 54, 55, 63, 64, 65, 66, 68, 69, 90, 91, and 93.

[CN12] Yuanmi Chen and Phong Q. Nguyen. Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In Pointcheval and Johansson [PJ12], pages 502–519.
→ Cited on pages 80, 81, 84, 85, 86, 92, 98, 104, 112, 150, and 155.

[CN13] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. 2013. Full version. Available at `http://www.di.ens.fr/~ychen/research/Full_BKZ.pdf`.
→ Cited on pages 8, 24, and 91.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In Pointcheval and Johansson [PJ12], pages 446–464.
→ Cited on pages 1, 4, 5, 73, 76, 77, 78, 80, 81, 84, 86, 91, 96, 98, 101, 103, 104, 105, 106, 107, 108, 110, 112, 113, 116, 117, 119, 120, 121, 123, and 129.

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997.
→ Cited on pages 81 and 87.

[Cor13] Jean-Sébastien Coron. Towards practical obfuscation of the AES, 2013. Private Slides.
→ Cited on pages 7 and 180.

[CPS13] David Cadé, Xavier Pujol, and Damien Stehlé. *fplll*, 4.0.4 edition, 2013. http://perso.ens-lyon.fr/damien.stehle/fplll/.
→ Cited on pages 91, 92, and 93.

[CS87] J. Chidambaraswamy and R. Sitaramachandrarao. On the probability that the values of m polynomials have a given g.c.d. *Journal of Number Theory*, 26:237–245, 1987.
→ Cited on page 86.

[CT12] Jean-Sébastien Coron and Mehdi Tibouchi. *An implementation of the DGHV fully homomorphic scheme*, 2012. Available under the GNU General Public License version 2 at `https://github.com/coron/fhe`.
→ Cited on pages 73, 76, 77, and 121.

[DDLL13a] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Canetti and Garay [CG13a], pages 40–56.
→ Cited on pages v, 2, 3, 9, 17, 31, 35, 40, 41, and 53.

[DDLL13b] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. *IACR Cryptology ePrint Archive*, 2013:383, 2013.
→ Cited on pages 9, 17, 31, 41, and 53.

[DG14] Nagarjun C. Dwarakanath and Steven D. Galbraith. Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. *Applicable Algebra in Engineering, Communication and Computing*, pages 1–22, 2014.
→ Cited on pages 32, 33, 34, 39, and 40.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
→ Cited on pages 11, 12, and 160.

[DL13] Léo Ducas and Tancrède Lepoint. *A Proof-of-concept Implementation of BLISS*, 2013. Available under the CeCILL License at `http://bliss.di.ens.fr`.
→ Cited on pages 9, 17, 39, 40, 53, 59, 69, and 71.

[DLPR13a] Cécile Delerablée, Tancrède Lepoint, Pascal Paillier, and Matthieu Rivain. White-box security notions for symmetric encryption schemes. *IACR Cryptology ePrint Archive*, 2013:523, 2013.
→ Cited on pages v, 8, 9, and 19.

[DLPR13b] Cécile Delerablée, Tancrède Lepoint, Pascal Paillier, and Matthieu Rivain. White-box security notions for symmetric encryption schemes. In Lange et al. [LLL13].
→ Cited on pages 9, 19, and 180.

[DN12a] Léo Ducas and Phong Q. Nguyen. Faster Gaussian lattice sampling using lazy floating-point arithmetic. In Wang and Sako [WS12], pages 415–432.
→ Cited on pages 2, 16, 31, and 175.

[DN12b] Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In Wang and Sako [WS12], pages 433–450.
→ Cited on pages 41, 55, and 64.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Safavi-Naini and Canetti [SNC12], pages 643–662.
→ Cited on page 41.

[DSES14] Yarkin Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. Toward practical homomorphic evaluation of block ciphers using Prince, 2014. WAHC'14 - 2nd Workshop on Applied Homomorphic Cryptography and Encrypted Computing.
→ Cited on pages 8, 137, and 176.

[DT14] Jintai Ding and Chengdong Tao. A new algorithm for solving the approximate common divisor problem and cryptanalysis of the FHE based on GACD. *IACR Cryptology ePrint Archive*, 2014:42, 2014.
→ Cited on page 90.

[Duc13] Léo Ducas. *Signatures Fondées sur les Réseaux Euclidiens: Attaques, Analyse et Optimisations*. PhD thesis, Université Paris Diderot, 2013.
→ Cited on pages 3, 32, 39, and 174.

[ECR12] ECRYPT II. ECRYPT II yearly report on algorithms and keysizes (2011-2012). Available on http://www.ecrypt.eu.org/, 2012.
→ Cited on pages 52, 69, 71, 81, and 91.

[Fei03] Joan Feigenbaum, editor. *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, volume 2696 of *Lecture Notes in Computer Science*. Springer, 2003.
→ Cited on pages 189 and 196.

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Canetti and Garay [CG13a], pages 513–530.
→ Cited on page 139.

[FIP01] Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.
→ Cited on pages 12, 15, 74, 130, and 135.

[FIP12] Secure hash standard (SHS). Federal Information Processing Standards Publication 180-4, 2012. http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf.
→ Cited on page 15.

[FS96] Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In Maurer [Mau96], pages 245–255.
→ Cited on page 58.

[FSF+13] Simon Fau, Renaud Sirdey, Caroline Fontaine, Carlos Aguilar Melchor, and Guy Gogniat. Towards practical program execution over fully homomorphic encryption schemes. In Fatos Xhafa, Leonard Barolli, Dritan Nace, Salvatore Venticinque, and Alain Bui, editors, *3PGCIC*, pages 284–290. IEEE, 2013.
→ Cited on pages 73 and 77.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.
→ Cited on pages 8, 32, 73, 77, 78, 119, 123, and 137.

[Gab13] Philippe Gaborit, editor. *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings*, volume 7932 of *Lecture Notes in Computer Science*. Springer, 2013.
→ Cited on pages 194 and 198.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.
→ Cited on pages 3, 4, 7, 15, 16, 29, 73, 75, 76, 94, 97, 99, 119, 120, 121, 123, 160, 162, and 175.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer, 1997.
→ Cited on pages 2, 31, and 41.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Johansson and Nguyen [JN13], pages 1–17.
→ Cited on pages 1, 2, 7, 13, 15, 18, 29, 139, 140, 141, 142, 143, 144, 145, 150, 151, 152, 156, 157, 159, 160, 161, 162, 164, 176, and 180.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all

circuits. In *FOCS*, pages 40–49. IEEE Computer Society, 2013.
→ Cited on pages 1, 4, 7, 15, 19, 29, 139, 176, and 180.

[GGH+13c] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Canetti and Garay [CG13b], pages 479–499.
→ Cited on page 140.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94. Springer, 2014.
→ Cited on page 139.

[GH11a] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Ostrovsky [Ost11], pages 107–109.
→ Cited on page 73.

[GH11b] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In Paterson [Pat11], pages 129–148.
→ Cited on pages 1, 6, 73, 76, 77, 120, 160, 162, and 175.

[GHM05] Judy Goldsmith, Matthias Hagen, and Martin Mundhenk. Complexity of DNF and isomorphism of monotone formulas. In Joanna Jedrzejowicz and Andrzej Szepietowski, editors, *MFCS*, volume 3618 of *Lecture Notes in Computer Science*, pages 410–421. Springer, 2005.
→ Cited on pages 5 and 122.

[GHPS13] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. Field switching in BGV-style homomorphic encryption. *Journal of Computer Security*, 21(5):663–684, 2013.
→ Cited on page 73.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012.
→ Cited on page 73.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In Pointcheval and Johansson [PJ12], pages 465–482.
→ Cited on pages 73, 77, 79, 80, 101, and 129.

[GHS12c] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Safavi-Naini and Canetti [SNC12], pages 850–867.
→ Cited on pages 1, 5, 6, 73, 74, 77, 107, 126, 129, 130, 135, 137, and 175.

[Gil10] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.
→ Cited on pages 194 and 206.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *STOC*, pages 25–32. ACM, 1989.
→ Cited on page 66.

[GLN12] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML confidential: Machine learning on encrypted data. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC*, volume 7839 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2012.
→ Cited on pages 6 and 175.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 530–547. Springer, 2012.
→ Cited on pages 4, 29, 32, 41, 52, 53, 55, 59, 60, 64, 67, 68, 69, and 175.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *STOC*, pages 365–377. ACM, 1982.
→ Cited on page 75.

[GM03] Daniel Goldstein and Andrew Mayer. On the equidistribution of Hecke points. *Forum Mathematicum*, 15:165–189, 2003.
→ Cited on page 92.

[GMPS14] Sourav Sen Gupta, Subhamoy Maitra, Goutam Paul, and Santanu Sarkar. (Non-)random sequences from (non-)random permutations - analysis of RC4 stream cipher. *J. Cryptology*, 27(1):67–108, 2014.
→ Cited on page 173.

[GMQ07] Louis Goubin, Jean-Michel Masereel, and Michaël Quisquater. Cryptanalysis of white box DES implementations. In Adams et al. [AMW07], pages 278–295.
→ Cited on pages 8 and 179.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 2008.
→ Cited on pages 23, 54, 63, 64, 66, and 90.

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Gilbert [Gil10], pages 257–278.
→ Cited on pages 24, 64, and 91.

[GOPS13] Tim Güneysu, Tobias Oder, Thomas Pöppelmann, and Peter Schwabe. Software speed records for lattice-based signatures. In Gaborit [Gab13], pages 67–82.
→ Cited on pages 29, 41, and 53.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *STOC*, pages 197–206. ACM, 2008.
→ Cited on pages 31, 33, 36, 39, and 41.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Vadhan [Vad07], pages 194–213.
→ Cited on pages 7, 139, and 180.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Miller [Mil96], pages 212–219.
→ Cited on page 24.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Canetti and Garay [CG13a], pages 75–92.
→ Cited on pages 73, 77, and 176.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Boneh et al. [BRF13], pages 545–554.
→ Cited on pages 15, 26, and 29.

[H20] ICT 2014 - information and communications technologies. `http://ec.europa.eu/research/participants/portal/desktop/en/opportunities/h2020/topics/96-ict-32-2014.html`, accessed 3 June 2014.
→ Cited on pages 6 and 176.

[HG01] Nick Howgrave-Graham. Approximate integer common divisors. In Silverman [Sil01], pages 51–66.
→ Cited on pages 4, 80, 81, 84, 87, and 105.

[HG07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, 2007.
→ Cited on pages 54, 55, 64, 68, and 69.

[HHGP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 122–140. Springer, 2003.
→ Cited on pages 3, 31, 32, 41, 44, 52, 64, and 65.

[HHGPW10] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, and William Whyte. Practical lattice-based cryptography: NTRUEncrypt and NTRUSign. In Nguyen and Vallée [NV10], pages 349–390.
→ Cited on page 54.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
→ Cited on page 26.

[HM08] Mathias Herrmann and Alexander May. Solving linear equations modulo divisors: On factoring given any bits. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 406–424. Springer, 2008.
→ Cited on page 156.

[HP07] John L. Hennessy and David A. Patterson. *Computer Architecture - A Quantitative Approach (4. ed.)*. Morgan Kaufmann, 2007.
→ Cited on page 77.

[HPP06] Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors. *Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings*, volume 4076 of *Lecture Notes in Computer Science*. Springer, 2006.
→ Cited on pages 201 and 207.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.
→ Cited on pages 2, 3, 16, 29, 32, 44, 52, 53, 54, 71, and 174.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Rogaway [Rog11], pages 447–464.
→ Cited on pages 24, 64, and 91.

[HPS+13] Jeffrey Hoffstein, Jill Pipher, John Schanck, Joseph H. Silverman, and William Whyte. Practical signatures from the partial Fourier recovery problem. *IACR Cryptology ePrint Archive*, 2013:757, 2013.
→ Cited on page 72.

[HRSV07] Susan Hohenberger, Guy N. Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In Vadhan [Vad07], pages 233–252.
→ Cited on page 179.

[HS13] Shai Halevi and Victor Shoup. *HElib*, 2013. Available under GNU General Public License (GPL) at `https://github.com/shaih/HElib`.
→ Cited on pages 73 and 77.

[HSW13a] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Canetti and Garay [CG13a], pages 494–512.
→ Cited on page 139.

[HSW13b] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2013:509, 2013.
→ Cited on page 139.

[IEE08] IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. *IEEE P1363.1-2008*, 2008.
→ Cited on pages 41 and 174.

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Vadhan [Vad07], pages 575–594.
→ Cited on page 75.

[JBF02] Matthias Jacob, Dan Boneh, and Edward W. Felten. Attacking an obfuscated cipher by injecting faults. In Feigenbaum [Fei03], pages 16–31.
→ Cited on pages 8 and 179.

[JL11] Marc Joye and Tancrède Lepoint. Traitor tracing schemes for protected software implementations. In Yan Chen, Stefan Katzenbeisser, and Ahmad-Reza Sadeghi, editors, *Digital Rights Management Workshop*, pages 15–22. ACM, 2011.
→ Cited on pages 9, 16, and 17.

[JL12] Marc Joye and Tancrède Lepoint. Partial key exposure on RSA with private exponents larger than N. In Mark Dermot Ryan, Ben Smyth, and Guilin Wang, editors, *ISPEC*, volume 7232 of *Lecture Notes in Computer Science*, pages 369–380. Springer, 2012.
→ Cited on pages 9, 16, and 17.

[JN13] Thomas Johansson and Phong Q. Nguyen, editors. *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*. Springer, 2013.
→ Cited on pages 17, 75, 129, 189, 192, and 199.

[Jou00] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In Wieb Bosma, editor, *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.
→ Cited on pages 6, 12, 15, 139, 159, 160, and 176.

[Joy09] Marc Joye. Basics of side-channel analysis. In Çetin Kaya Koç [cKK09], pages 365–380.
→ Cited on page 14.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *STOC*, pages 193–206. ACM, 1983.
→ Cited on pages 24, 64, and 91.

[Kar10] Mohamed Karroumi. Protecting white-box AES with dual ciphers. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC*, volume 6829 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 2010.
→ Cited on pages 8 and 179.

[Ker83] Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, pages 161–191, 1883.
→ Cited on page 11.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *STOC*, pages 20–31. ACM, 1988.
→ Cited on page 180.

[Kil01] Joe Kilian, editor. *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*. Springer, 2001.
→ Cited on page 187.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In David B. Shmoys, editor, *SODA*, pages 937–941. ACM/SIAM, 2000.
→ Cited on page 31.

[KLYC13] Jinsu Kim, Moon Sung Lee, Aaram Yun, and Jung Hee Cheon. CRT-based fully homomorphic encryption over the integers. *IACR Cryptology ePrint Archive*, 2013:57, 2013.
→ Cited on pages 75, 95, 97, and 142.

[Kra14] Hugo Krawczyk, editor. *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*. Springer, 2014.
→ Cited on pages 17, 75, 107, 129, 190, and 198.

[KS09] Emilia Käsper and Peter Schwabe. Faster and timing-attack resistant AES-GCM. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2009.
→ Cited on pages 129, 130, and 133.

[Lag82] J. C. Lagarias. The computational complexity of simultaneous diophantine approximation problems. In *FOCS*, pages 32–39. IEEE Computer Society, 1982.
→ Cited on page 88.

[Lana] Adam Langley. ChaCha20 and Poly1305 for TLS. `https://www.imperialviolet.org/2013/10/07/chacha20.html`, accessed 3 June 2014.
→ Cited on page 173.

[Lanb] Adam Langley. TLS symmetric crypto. `https://www.imperialviolet.org/2014/02/27/tlssymmetriccrypto.html`, accessed 3 June 2014.
→ Cited on page 173.

[Len87] Hendrik W. Lenstra Jr. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.
→ Cited on page 81.

[Lep13] Tancrède Lepoint. *An Implementation of Multilinear Maps over the Integers*, 2013. Available under the Creative Commons License BY-NC-SA at `https://github.com/tlepoint/multimap`.
→ Cited on pages 9, 18, 159, 160, 163, 164, and 176.

[Lep14] Tancrède Lepoint. *A proof-of-concept implementation of the homomorphic evaluation of SIMON using FV and YASHE leveled homomorphic cryptosystems*, 2014. Available under the CeCILL License at `https://github.com/tlepoint/homomorphic-simon`.
→ Cited on pages 9, 18, 73, and 77.

[LJMP90] Arjen K. Lenstra, Hendrik W. Lenstra Jr., Mark S. Manasse, and John M. Pollard. The number field sieve. In Harriet Ortiz, editor, *STOC*, pages 564–572. ACM, 1990.
→ Cited on pages 13 and 81.

[LLL82] Arjen K. Lenstra, Hendrick W. Lenstra Jr., and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.
→ Cited on pages 2, 23, and 90.

[LLL13] Tanja Lange, Kristin Lauter, and Petr Lisonek, editors. *SAC*, Lecture Notes in Computer Science. Springer, 2013.
→ Cited on pages 19, 186, 191, and 199.

[LLLS13] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In Sako and Sarkar [SS13], pages 41–61.
→ Cited on page 72.

[LLNW14] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In Krawczyk [Kra14], pages 345–361.
→ Cited on page 72.

[LLS14] Fabien Laguillaumie, Adeline Langlois, and Damien Stehlé. Chiffrement avancé à partir du problème Learning with errors. In Sylvain Peyronnet, editor, *Informatique Mathématique une photographie en 2014*, pages 179–225. Presses Universitaires de Perpignan, 2014.
→ Cited on pages 15, 16, 25, and 26.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2006.
→ Cited on page 25.

[LMP13] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Solving the shortest vector problem in lattices faster using quantum search. In Gaborit [Gab13], pages 83–101.
→ Cited on page 24.

[LMSV11] Jake Loftus, Alexander May, Nigel P. Smart, and Frederik Vercauteren. On CCA-secure somewhat homomorphic encryption. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 55–72. Springer, 2011.
→ Cited on page 73.

[LN05] Hamilton E. Link and William D. Neumann. Clarifying obfuscation: Improving the security of white-box DES. In *ITCC (1)*, pages 679–684. IEEE Computer Society, 2005.
→ Cited on pages 8 and 179.

[LN14a] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In Pointcheval and Vergnaud [PV14], pages 318–335.
→ Cited on pages v, 6, 8, 9, 18, 73, 91, 137, 174, 175, and 176.

[LN14b] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. *IACR Cryptology ePrint Archive*, 2014:62, 2014.
→ Cited on pages 9 and 18.

[LP13] Tancrède Lepoint and Pascal Paillier. On the minimal number of bootstrappings in homomorphic circuits. In Adams et al. [ABS13], pages 189–200.
→ Cited on pages v, 5, 6, 9, 18, 73, and 119.

[LPR13a] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013.
→ Cited on pages 25, 26, 29, and 174.

[LPR13b] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for Ring-LWE cryptography. In Johansson and Nguyen [JN13], pages 35–54.
→ Cited on page 26.

[LR13] Tancrède Lepoint and Matthieu Rivain. Another nail in the coffin of white-box AES implementations. *IACR Cryptology ePrint Archive*, 2013:455, 2013.
→ Cited on pages 9 and 19.

[LRM+13] Tancrède Lepoint, Matthieu Rivain, Yoni De Mulder, Peter Roelse, and Bart Preneel. Two attacks on a white-box AES implementation. In Lange et al. [LLL13].
→ Cited on pages v, 8, 9, 19, 179, and 180.

[LS12] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. Cryptology ePrint Archive, Report 2012/090, 2012. `http://eprint.iacr.org/`. To appear in Designs, Codes and Cryptography.
→ Cited on page 26.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Nguyen and Oswald [NO14], pages 239–256.
→ Cited on pages 142, 143, 157, and 164.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *STOC*, pages 1219–1234. ACM, 2012.
→ Cited on pages 7, 54, 73, 77, 78, 119, and 123.

[Lud03] Christoph Ludwig. A faster lattice reduction method using quantum search. In Toshihide Ibaraki, Naoki Katoh, and Hirotaka Ono, editors, *ISAAC*, volume 2906 of *Lecture Notes in Computer Science*, pages 199–208. Springer, 2003.
→ Cited on page 24.

[Lyu] Vadim Lyubashevsky. Lattice-based encryption. `http://www.di.ens.fr/~lyubash/talks/LWEcrypto.pdf`, accessed 3 June 2014.
→ Cited on page 174.

[Lyu08] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In Ronald Cramer, editor, *Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2008.
→ Cited on pages 2 and 41.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.
→ Cited on pages 2 and 41.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In Pointcheval and Johansson [PJ12], pages 738–755.
→ Cited on pages 2, 3, 29, 32, 33, 41, 42, 43, 44, 47, 48, 51, 52, 53, 56, 58, 64, 67, and 69.

[Mau96] Ueli M. Maurer, editor. *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.
→ Cited on pages 192 and 203.

[MGH08] Wil Michiels, Paul Gorissen, and Henk D. L. Hollmann. Cryptanalysis of a generic class of white-box implementations. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 414–428. Springer, 2008.
→ Cited on pages 8 and 179.

[MHM+13] Ciara Moore, Neil Hanley, John McAllister, Máire O'Neill, Elizabeth O'Sullivan, and Xiaolin Cao. Targeting FPGA DSP slices for a large integer multiplier for integer based FHE. In Adams et al. [ABS13], pages 226–237.
→ Cited on pages 73 and 77.

[Mic10] Daniele Micciancio. A first glimpse of cryptography's Holy Grail. *Commun. ACM*, 53(3):96, 2010.
→ Cited on pages 1, 4, 16, 75, and 139.

[Mil96] Gary L. Miller, editor. *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. ACM, 1996.
→ Cited on pages 185 and 194.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Rogaway [Rog11], pages 465–484.
→ Cited on pages 64, 66, and 69.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval and Johansson [PJ12], pages 700–718.
→ Cited on pages 31 and 41.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Canetti and Garay [CG13a], pages 21–39.
→ Cited on page 26.

[MPSW09] Tal Malkin, Chris Peikert, Rocco A. Servedio, and Andrew Wan. Learning an overcomplete basis: Analysis of lattice-based signatures with perturbations, 2009. Manuscript. Available in [Wan10, Chapter 6].
→ Cited on pages 41, 55, and 64.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.
→ Cited on pages 25 and 33.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer Berlin Heidelberg, 2009.
→ Cited on pages 26, 29, 64, 66, and 69.

[MRP12] Yoni De Mulder, Peter Roelse, and Bart Preneel. Cryptanalysis of the Xiao - Lai white-box AES implementation. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2012.
→ Cited on pages 8 and 179.

[MWP10] Yoni De Mulder, Brecht Wyseur, and Bart Preneel. Cryptanalysis of a perturbated white-box AES implementation. In Guang Gong and Kishan Chand Gupta, editors, *INDOCRYPT*, volume 6498 of *Lecture Notes in Computer Science*, pages 292–310. Springer, 2010.
→ Cited on pages 8 and 179.

[NIS11] NIST Special Publication 800-131A. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. Available on `http://csrc.nist.gov`, 2011.
→ Cited on pages 52, 69, 71, and 91.

[NIS12] NIST. SHA-3 competition, 2007-2012. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html.
→ Cited on pages 11 and 173.

[NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, *CCSW*, pages 113–124. ACM, 2011.
→ Cited on pages 6, 73, 77, 129, and 175.

[NO14] Phong Q. Nguyen and Elisabeth Oswald, editors. *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*. Springer, 2014.
→ Cited on pages 187 and 199.

[NR09] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009.
→ Cited on pages 41, 55, and 64.

[NS99] Phong Q. Nguyen and Jacques Stern. The hardness of the hidden subset sum problem and its cryptographic implications. In Wiener [Wie99], pages 31–46.
→ Cited on page 155.

[NS01] Phong Q. Nguyen and Jacques Stern. The two faces of lattices in cryptology. In Silverman [Sil01], pages 146–180.
→ Cited on pages 15 and 89.

[NS05] Phong Q. Nguyen and Damien Stehlé. Floating-point LLL revisited. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233. Springer, 2005.
→ Cited on page 92.

[NS06] Phong Q. Nguyen and Damien Stehlé. LLL on the average. In Hess et al. [HPP06], pages 238–256.
→ Cited on pages 90, 91, and 92.

[NSV11] Andrew Novocin, Damien Stehlé, and Gilles Villard. An LLL-reduction algorithm with quasi-linear time complexity: extended abstract. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 403–412. ACM, 2011.
→ Cited on page 93.

[NV10] Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL Algorithm - Survey and Applications*. Information Security and Cryptography. Springer, 2010.
→ Cited on pages 195, 203, and 205.

[Odl90] Andrew M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Cryptology and Computational Number Theory*, volume 42 of *Proc. of Symposia in Applied Mathematics*, pages 75–88. AMS, 1990.
→ Cited on page 15.

[OPG14] Tobias Oder, Thomas Pöppelmann, and Tim Güneysu. Beyond ECDSA and RSA: Lattice-based digital signatures on constrained devices. In *DAC*. 2014.
→ Cited on pages 4, 29, 40, 71, and 175.

[Ost11] Rafail Ostrovsky, editor. *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. IEEE, 2011.
→ Cited on pages 188 and 193.

[OYKU10] Naoki Ogura, Go Yamamoto, Tetsutaro Kobayashi, and Shigenori Uchiyama. An improvement of key generation algorithm for Gentry's homomorphic encryption scheme. In Isao Echizen, Noboru Kunihiro, and Ryôichi Sasaki, editors, *IWSEC*, volume 6434 of *Lecture Notes in Computer Science*, pages 70–83. Springer, 2010.
→ Cited on pages 73 and 76.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
→ Cited on pages 75 and 175.

[Pat11] Kenneth G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.
→ Cited on pages 193 and 205.

[PBS11a] Henning Perl, Michael Brenner, and Matthew Smith. Poster: an implementation of the fully homomorphic Smart-Vercauteren crypto-system. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 837–840. ACM, 2011.
→ Cited on pages 73, 77, 120, and 121.

[PBS11b] Henning Perl, Michael Brenner, and Matthew Smith. *Scarab library*, 2011. Available under the MIT license at `https://hcrypt.com/scarab-library/`.
→ Cited on pages 73, 77, and 121.

[PDG14] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. *IACR Cryptology ePrint Archive*, 2014:254, 2014. To appear at CHES 2014.
→ Cited on page 175.

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 2010.
→ Cited on pages 31, 33, and 39.

[PG12] Thomas Pöppelmann and Tim Güneysu. Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In Alejandro Hevia and Gregory Neven, editors, *LATINCRYPT*, volume 7533 of *Lecture Notes in Computer Science*, pages 139–158. Springer, 2012.
→ Cited on pages 4, 58, and 175.

[PG13] Thomas Pöppelmann and Tim Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. 2013. SAC 2013.
→ Cited on pages 4, 29, and 175.

[PG14] Thomas Pöppelmann and Tim Güneysu. Area optimization of lightweight lattice-based encryption on reconfigurable hardware. 2014.
→ Cited on pages 4, 29, and 175.

[PJ12] David Pointcheval and Thomas Johansson, editors. *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*. Springer, 2012.
→ Cited on pages 190, 193, 199, and 200.

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 145–166. Springer, 2006.
→ Cited on page 25.

[Pri51] G. Baley Price. Bounds for determinants with dominant principal diagonal. *Proceedings of the American Mathematical Society*, 2(3):497–502, 1951.
→ Cited on page 154.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Maurer [Mau96], pages 387–398.
→ Cited on page 45.

[PS14] David Pointcheval and Olivier Sanders. Forward secure non-interactive key exchange, 2014. Manuscript.
→ Cited on page 140.

[PTT10] Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Optimal authenticated data structures with multilinear forms. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *Pairing*, volume 6487 of *Lecture Notes in Computer Science*, pages 246–264. Springer, 2010.
→ Cited on pages 139 and 141.

[PV14] David Pointcheval and Damien Vergnaud, editors. *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, volume 8469 of *Lecture Notes in Computer Science*. Springer, 2014.
→ Cited on pages 18 and 198.

[RAD78] Ronald L. Rivest, Leonard M. Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.
→ Cited on pages 4, 73, and 75.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.
→ Cited on pages 3, 5, 15, 25, 107, and 174.

[Reg10a] Oded Regev. Learning with errors over rings. In Guillaume Hanrot, François Morain, and Emmanuel Thomé, editors, *ANTS*, volume 6197 of *Lecture Notes in Computer Science*, page 3. Springer, 2010.
→ Cited on page 26.

[Reg10b] Oded Regev. On the complexity of lattice problems with polynomial approximation factors. In Nguyen and Vallée [NV10], pages 475–496.
→ Cited on page 25.

[Rog11] Phillip Rogaway, editor. *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*. Springer, 2011.
→ Cited on pages 188, 190, 195, and 200.

[Roh09] Pankaj Rohatgi. Improved techniques for side-channel analysis. In Çetin Kaya Koç [cKK09], pages 381–406.
→ Cited on page 14.

[Rot13] Ron Rothblum. On the circular security of bit-encryption. In *TCC*, pages 579–598, 2013.
→ Cited on pages 139 and 143.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
→ Cited on page 132.

[RS09] Markus Rückert and Dominique Schröder. Aggregate and verifiably encrypted signatures from multilinear maps without random oracles. In Jong Hyuk Park, Hsiao-Hwa Chen, Mohammed Atiquzzaman, Changhoon Lee, Tai-Hoon Kim, and Sang-Soo Yeo, editors, *ISA*, volume 5576 of *Lecture Notes in Computer Science*, pages 750–759. Springer, 2009.
→ Cited on page 139.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
→ Cited on page 13.

[Rüc10] Markus Rückert. Lattice-based blind signatures. In Abe [Abe10], pages 413–430.
→ Cited on page 41.

[RVV13] Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. High precision discrete Gaussian sampling on FPGAs. 2013.
→ Cited on pages 34 and 40.

[RWC] Real world cryptography workshop. `http://www.realworldcrypto.com/`, accessed 3 June 2014.
→ Cited on page 174.

[S+14] W. A. Stein et al. *Sage Mathematics Software (Version 6.1.1)*. The Sage Development Team, 2014. `http://www.sagemath.org`.
→ Cited on pages 76, 86, 91, 104, 106, and 174.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.
→ Cited on pages 2 and 42.

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994.
→ Cited on pages 2 and 23.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
→ Cited on page 15.

[Sil01] Joseph H. Silverman, editor. *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.
→ Cited on pages 195 and 201.

[SNC12] Reihaneh Safavi-Naini and Ran Canetti, editors. *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*. Springer, 2012.
→ Cited on pages 188, 191, and 193.

[SOK00] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, 2000.
→ Cited on pages 6, 15, 139, and 176.

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Abe [Abe10], pages 377–394.
→ Cited on pages 73 and 76.

[SS11a] Peter Scholl and Nigel P. Smart. Improved key generation for Gentry's fully homomorphic encryption scheme. In Liqun Chen, editor, *IMA Int. Conf.*, volume 7089 of *Lecture Notes in Computer Science*, pages 10–22. Springer, 2011.
→ Cited on pages 73 and 76.

[SS11b] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In Paterson [Pat11], pages 27–47.
→ Cited on pages 54 and 174.

[SS13] Kazue Sako and Palash Sarkar, editors. *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, volume 8270 of *Lecture Notes in Computer Science*. Springer, 2013.
→ Cited on pages 188 and 198.

[ST] Nigel P. Smart and Stefan Tillich. Circuits of basic functions suitable for MPC and FHE. `http://www.cs.bris.ac.uk/Research/CryptographySecurity/MPC/`, accessed 12 May 2014.
→ Cited on pages 121, 125, and 126.

[Sta13] Martijn Stam, editor. *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, volume 8308 of *Lecture Notes in Computer Science*. Springer, 2013.
→ Cited on pages 187 and 206.

[Ste10] Damien Stehlé. Floating-point LLL: Theoretical and practical aspects. In Nguyen and Vallée [NV10], pages 179–213.
→ Cited on pages 92 and 93.

[Ste11] Damien Stehlé. *Euclidean Lattices: Algorithms and Cryptography.* Mémoire d'habilitation à diriger des recherches, École Normale Supérieure de Lyon, October 2011.
→ Cited on page 16.

[Ste13] Marc Stevens. Counter-cryptanalysis. In Canetti and Garay [CG13a], pages 129–146.
→ Cited on page 173.

[Sto96] Arne Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. In Erwin Engeler, B. F. Caviness, and Yagati N. Lakshman, editors, *ISSAC*, pages 267–274. ACM, 1996.
→ Cited on page 157.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.
→ Cited on pages 73, 76, 77, and 79.

[SV11] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *IACR Cryptology ePrint Archive*, 2011:133, 2011.
→ Cited on pages 73, 77, and 79.

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *IACR Cryptology ePrint Archive*, 2013:454, 2013.
→ Cited on page 139.

[Vad07] Salil P. Vadhan, editor. *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*. Springer, 2007.
→ Cited on pages 194, 195, and 196.

[Var75] James M. Varah. A lower bound for the smallest singular value of a matrix. *Linear Algebra and its Applications*, 11(1):3–5, 1975.
→ Cited on page 154.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [Gil10], pages 24–43.
→ Cited on pages 1, 4, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 84, 85, 87, 88, 89, 93, 94, 96, 97, 98, 99, 100, 106, 107, 108, 109, 119, 120, 121, and 123.

[vdPS13] Joop van de Pol and Nigel P. Smart. Estimating key sizes for high dimensional lattice-based systems. In Stam [Sta13], pages 290–303.
→ Cited on pages 8 and 91.

[vN51] John von Neumann. Various techniques used in connection with random digits. *J. Research Nat. Bur. Stand., Appl. Math. Series*, 12:36–38, 1951.
→ Cited on pages 2 and 27.

[Wan10] Andrew Wan. *Learning, Cryptography, and the Average Case*. PhD thesis, Columbia University, 2010.
→ Cited on page 200.

[WEJ13] William Whyte, Mark Etzel, and Peter Jenney. *Open Source NTRU Public Key Cryptography Algorithm and Reference Code*, 2013. Available under the Gnu Public License (GPL) at `https://github.com/NTRUOpenSourceProject/ntru-crypto`.
→ Cited on page 71.

[Wie99] Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.
→ Cited on pages 186 and 201.

[WMGP07] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of white-box DES implementations with arbitrary external encodings. In Adams et al. [AMW07], pages 264–277.
→ Cited on pages 8 and 179.

[WP05] Brecht Wyseur and Bart Preneel. Condensed white-box implementations. *Proceedings of the 26th Symposium on Information Theory in the Benelux*, pages 296–301, 2005.
→ Cited on pages 8 and 179.

[WS12] Xiaoyun Wang and Kazue Sako, editors. *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*. Springer, 2012.
→ Cited on pages 186 and 191.

[WT99] Alma Whitten and J Doug Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium*, volume 99. McGraw-Hill, 1999.
→ Cited on page 73.

[XL09] Yaying Xiao and Xuejia Lai. A Secure Implementation of White-Box AES. In *CSA 2009*, pages 1–6, 2009.
→ Cited on pages 8 and 179.

[YH13] Eric A. Young and Tim J. Hudson. *Open Source Secure Sockets Layer (Version 1.0.1c)*, 2013. `http://www.openssl.org/`.
→ Cited on page 53.

[ZD06] Paul Zimmermann and Bruce Dodson. 20 years of ECM. In Hess et al. [HPP06], pages 525–542.
→ Cited on page 81.

Abstract.

Today, lattice-based cryptography is a thriving scientific field. Its swift expansion is due, among others, to the attractiveness of fully homomorphic encryption and cryptographic multilinear maps. Lattice-based cryptography has also been recognized for its thrilling properties: a security that can be reduced to worst-case instances of problems over lattices, a quasi-optimal asymptotic efficiency and an alleged resistance to quantum computers. However, its practical use in real-world products leaves a lot to be desired. This thesis accomplishes a step towards this goal by narrowing the gap between theoretical research and practical implementation of recent public key cryptosystems.
In this thesis, we design and implement a lattice-based digital signature, two fully homomorphic encryption schemes and cryptographic multilinear maps.
Our highly efficient signature scheme, BLISS, opened the way to implementing lattice-based cryptography on constrained devices and remains as of today a promising primitive for post-quantum cryptography. Our fully homomorphic encryption schemes enjoy competitive homomorphic evaluations of nontrivial circuits. Finally, we describe the first implementation of cryptographic multilinear maps. Based on our implementation, a non interactive key exchange between more than three parties has been realized for the first time, and amounts to a few seconds per party.

Keywords: public key cryptography, lattices, digital signature, fully homomorphic encryption, multilinear maps, implementation.

Résumé.

La cryptographie à base de réseaux euclidiens est aujourd'hui un domaine scientifique en pleine expansion et connait une évolution rapide et accélérée par l'attractivité du chiffrement complètement homomorphe ou des applications multilinéaires cryptographiques. Ses propriétés sont très attractives : une sécurité pouvant être réduite à la difficulté des pires cas de problèmes sur les réseaux euclidiens, une efficacité asymptotique quasi-optimale et une résistance présupposée aux ordinateurs quantiques. Cependant, on dénombre encore peu de résultats de recherche sur les constructions à visée pratique pour un niveau de sécurité fixé. Cette thèse s'inscrit dans cette direction et travaille à réduire l'écart entre la théorie et la pratique de la cryptographie à clé publique récente.
Dans cette thèse, nous concevons et implémentons une signature numérique basée sur les réseaux euclidiens, deux schémas de chiffrement complètement homomorphe et des applications multilinéaires cryptographiques.
Notre signature digitale ultra-performante, BLISS, ouvre la voie à la mise en pratique de la cryptographie à base de réseaux sur petites architectures et est un candidat sérieux à la cryptographie post-quantique. Nos schémas de chiffrement complètement homomorphes permettent d'évaluer des circuits non triviaux de manière compétitive. Finalement, nous proposons la première implémentation d'applications multilinéaires et réalisons, pour la première fois, un échange de clé non interactif entre plus de trois participants en quelques secondes.

Mots clés : cryptographie à clé publique, réseaux euclidiens, signature numérique, chiffrement homomorphe, applications multilinéaires, implémentation.