Numéro National de Thèse : 2019LYSEN048

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée par
l'École Normale Supérieure de Lyon

École Doctorale N°512
École Doctorale en Informatique et Mathématiques de Lyon

Spécialité de doctorat : Informatique

Soutenue publiquement le 16/10/2019, par :
Alice Pellet--Mary

Réseaux idéaux et fonction multilinéaire GGH13

On ideal lattices and the GGH13 multilinear map

Devant le jury composé de :

Cheon Jung Hee, professeur, Université Nationale de Séoul (Corée)	Rapporteur
Pointcheval David, directeur de recherche, CNRS et ENS	Rapporteur
Vercauteren Frederik, professeur associé, KU Leuven (Belgique)	Rapporteur
Agrawal Shweta, professeur assistant, IIT Madras (Inde)	Examinatrice
Stehlé Damien, professeur, ENS de Lyon	Directeur de thèse

Résumé

La cryptographie à base de réseaux euclidiens est un domaine prometteur pour la construction de primitives cryptographiques post-quantiques. Un problème fondamental, lié aux réseaux, est le problème du plus court vecteur (ou SVP, pour Shortest Vector Problem). Ce problème est supposé être difficile à résoudre même avec un ordinateur quantique. Afin d'améliorer l'efficacité des protocoles cryptographiques, on peut utiliser des réseaux structurés, comme par exemple des réseaux idéaux ou des réseaux modules (qui sont une généralisation des réseaux idéaux). La sécurité de la plupart des schémas utilisant des réseaux structurés dépend de la difficulté du problème SVP dans des réseaux modules, mais un petit nombre de schémas peuvent également être impactés par SVP dans des réseaux idéaux. La principale construction pouvant être impactée par SVP dans des réseaux idéaux est la fonction multilinéaire GGH13. Cette fonction multilinéaire est principalement utilisée aujourd'hui pour construire des obfuscateurs de programmes, c'est-à-dire des fonctions qui prennent en entrée le code d'un programme et renvoie le code d'un programme équivalent (calculant la même fonction), mais qui doit cacher la façon dont le programme fonctionne.

Dans cette thèse, nous nous intéressons dans un premier temps au problème SVP dans les réseaux idéaux et modules. Nous présentons un premier algorithme qui, après un pre-calcul exponentiel, permet de trouver des vecteurs courts dans des réseaux idéaux plus rapidement que le meilleur algorithme connu pour des réseaux arbitraires. Nous présentons ensuite un algorithme pour les réseaux modules de rang 2, également plus efficace que le meilleur algorithme connu pour des réseaux arbitraires, à condition d'avoir accès à un oracle résolvant le problème du plus proche vecteur dans un réseau fixé. Ce deuxième algorithme peut ensuite être utilisé pour construire un algorithme LLL pour des modules de rang arbitraire. Le pré-calcul exponentiel et l'oracle pour le problème du plus proche vecteurs rendent ces algorithmes inutilisables en pratique.

Dans un second temps, nous nous intéressons à la fonction GGH13 ainsi qu'aux obfuscateurs qui l'utilisent. Nous étudions d'abord l'impact des attaques statistiques sur la fonction GGH13 et ses variantes. Nous nous intéressons ensuite à la sécurité des obfuscateurs utilisant la fonction GGH13 et proposons une attaque quantique contre plusieurs de ces obfuscateurs. Cette attaque quantique utilise entre autres un algorithme calculant un vecteur court dans un réseau idéal dépendant d'un paramètre secret de la fonction GGH13.

Abstract

Lattice-based cryptography is a promising area for constructing cryptographic primitives that are plausibly secure even in the presence of quantum computers. A fundamental problem related to lattices is the shortest vector problem (or SVP), which asks to find a shortest non-zero vector in a lattice. This problem is believed to be intractable, even quantumly. Structured lattices, for example ideal lattices or module lattices (the latter being a generalization of the former), are often used to improve the efficiency of lattice-based primitives. The security of most of the schemes based on structured lattices is related to SVP in module lattices, and a very small number of schemes can also be impacted by SVP in ideal lattices.

In this thesis, we first focus on the problem of finding short vectors in ideal and module lattices. We propose an algorithm which, after some exponential pre-computation, performs better on ideal lattices than the best known algorithm for arbitrary lattices. We also present an algorithm to find short vectors in rank 2 modules, provided that we have access to some oracle solving the closest vector problem in a fixed lattice. This second algorithm can then be used to construct an LLL algorithm for module lattices of arbitrary rank. The exponential pre-processing time and the oracle call make these algorithms unusable in practice.

The main scheme whose security might be impacted by SVP in ideal lattices is the GGH13 multilinear map. This protocol is mainly used today to construct program obfuscators, which should render the code of a program unintelligible, while preserving its functionality. In a second part of this thesis, we focus on the GGH13 map and its application to obfuscation. We first study the impact of statistical attacks on the GGH13 map and on its variants. We then study the security of obfuscators based on the GGH13 map and propose a quantum attack against multiple such obfuscators. This quantum attack uses as a subroutine an algorithm to find a short vector in an ideal lattice related to a secret element of the GGH13 map.

Remerciements / Acknowledgment

Je voudrais commencer de façon originale en remerciant mon directeur de thèse Damien Stehlé. Merci pour tous les conseils que tu m'as donnés, pour le temps que tu as passé à relire tout ce que j'écrivais (heureusement que tu fais ça sur tablette, sinon j'imagine que tu aurais une pile de stylos rouges morts dans un coin de ton bureau), et pour m'avoir fait découvrir plein de questions intéressantes auxquelles j'ai l'intention de continuer à réfléchir dans les années qui viennent.

I would also like to thank Jung Hee Cheon, David Pointcheval and Frederik Vercauteren for reviewing this thesis and providing me with useful comments. Thank you Frederik for welcoming me in Leuven for my post-doc, I am looking forward to it. Thank you also Shweta Agrawal for agreeing to come all the way from India to be in my jury, and for inviting me to India during my PhD.

Merci également à tous les membres (et anciens membres) de l'équipe AriC pour leur bonne humeur, les repas du midi sous pression et les longues pauses café : Bruno, Fabien, Paola, Guillaume, Vincent, Nathalie, Gilles, Alain, Benoît, Octavie, Fabrice, Chitchanok, Radu, Miruna, Ida, Huyen, Weiqiang, Florent, Elena, Alexandre, Changmin, Dingding, Gottfried, Hervé, Nicolas L., Claude-Pierre, Alonso, Junqing, Laurent, Anastasia, Jean-Michel, Nicolas B., Joris et Serge. A special thanks goes to my office-mates who put up with me and discussed with me when I didn't want to work: thank you Sanjay, Jiangtao, Alexandre, Alonso, Gottfried and Hervé. Thank you also to all the "young in Aric" for the occasional "gouter" with some food I never tried before (like a chili con carne without chili nor carne). Thank you Miruna and Elena for your bad luck when traveling with me and the nice evening we spent in Frankfurt "city center" on the way back from Bertinoro. Merci Ida de m'avoir tenu compagnie aux conseils de labo, et plus généralement d'être la bonne poire de service dès qu'on a besoin de quelqu'un pour faire quelque chose (par exemple relire ces remerciements). Comme je ne suis pas (trop) sectaire, j'aimerais également remercier des doctorants d'autres équipes, avec lesquels j'ai pu échanger sur la vie en thèse et partager des connaissances sur toutes les procédures administratives de réinscription ou de soutenance : merci Alexandre, Aurore, Laureline et Pierre. Enfin, je voudrais remercier Marie, Chiraz, Nelly, Kadiatou et Myriam pour avoir géré à ma place une bonne partie des désagréments administratifs et avoir toujours été disponibles quand j'avais des questions.

On the other side of the border, I would like to thank all the cryptology group of CWI for welcoming me during a very nice summer internship. Merci en particulier à Léo, Jessika et Benjamin pour les soirées flim (et le marché aux fromages). I would also like to thank my office-mates from there: dankjewel Koen, Wessel and Toon, it was *gezellig* to share an office with you.

Dans la catégorie "autres", je voudrais remercier Laure d'être restée en contact et de m'avoir invité à sa soutenance. Merci à Marc de m'avoir mis dans les remerciements de sa thèse. Merci à Alexandre Gélin d'avoir répondu à mes questions sur sa thèse sans (trop) m'insulter. Merci à Bon Papa pour les mardis gratins dauphinois et pour avoir toujours essayé de comprendre pourquoi je voulais m'o(b)fusquer. Merci à Clara pour les soirées cinéma/mexicain. Merci à Carine de m'avoir fait découvrir plein de restaurants sympas à Lyon. Et enfin, merci à mes parents et à mon frère, histoire de terminer ces remerciements de façon aussi originale que je les ai commencés.

Contents

Résumé long en français

L'un des principaux objectifs de la cryptographie est de garantir la confidentialité des messages, en les chiffrant durant les phases de transmission. Historiquement, les méthodes de chiffrement utilisaient des *clés symétriques secrètes*, partagées entre l'expéditeur et le destinataire. Ces schémas de chiffrement sont appelés *schémas de chiffrement symétriques*, et nécessitent une rencontre entre l'expéditeur et le destinataire (ou l'utilisation d'un tiers de confiance) avant de pouvoir communiquer en toute sécurité. Cette nécessité d'une rencontre entre l'expéditeur et le destinataire n'est pas raisonnable pour les applications actuelles de la cryptographie, par exemple sur Internet. Heureusement, Diffie et Hellman ont décrit en 1976 le premier schéma cryptographique à *clé publique* [DH76]. Dans un schéma à clé publique, chaque utilisateur possède sa propre clé secrète (qu'il ne doit partager avec personne), ainsi qu'une clé publique, qu'il peut diffuser publiquement. Cette clé publique permet à n'importe qui de chiffrer un message, cependant, seulement l'utilisateur possédant la clé secrète correspondante pourra déchiffrer le message et retrouver le texte d'origine. En d'autres termes, le clé publique et le message chiffré ne doivent révéler aucune information sur le message sous-jacent. Un schéma de chiffrement à clé publique peut être utilisé pour garantir la confidentialité d'une communication, sans que les personnes communiquant n'aient besoin de se rencontrer au préalable. En effet, le destinataire peut envoyer sa clé publique à l'expéditeur. L'expéditeur utilise la clé publique pour chiffrer son message, et renvoie le message chiffré au destinataire, qui pourra le déchiffrer grâce à sa clé secrète. Un attaquant écoutant la conversation ne connaîtra que la clé publique et le message chiffré, qui par définition ne doivent rien révéler du message transmis. Les protocoles cryptographiques de communication utilisés aujourd'hui utilisent en général un schéma de chiffrement à clé publique pour échanger une clé secrète symétrique entre les participants, puis un schéma de chiffrement symétrique utilisant cette clé secrète symétrique pour le reste de la conversation. Cette procédure est plus efficace que l'utilisation d'un schéma de chiffrement à clé publique pour toute la conversation, car les schémas symétriques sont en général plus efficaces que ceux à clé publique.

La sécurité des schémas à clé publique repose en général sur des hypothèses " simples", c'est-à-dire qui peuvent être décrites en quelques phrases. Plus précisément, sous l'hypothèse que certains problèmes mathématiques sont difficiles à résoudre, il est possible de prouver qu'aucun attaquant (pour un certain modèle d'attaquant) ne peut retrouver le message envoyé, s'il a seulement accès à la clé publique et au message chiffré. Le protocole de Diffie-Hellman mentionné ci-dessus repose sur un problème lié au logarithme discret. Étant donné un groupe G cyclique, un générateur de ce groupe g et un élément h arbitraire dans le groupe, le problème du logarithme discret consiste à calculer le logarithme de h en base g, c'est-à-dire l'entier α tel que $h = g^\alpha$. La difficulté supposée de ce problème a depuis été utilisée pour la construction de nombreuses autres primitives cryptographiques. Un autre problème de théorie des nombres très important en cryptographie est la factorisation. Étant donné un produit pq de deux grands nombres premiers p et q, l'objectif est de retrouver p et q. Un problème lié à la factorisation est utilisé pour évaluer la sécurité du chiffrement RSA [RSA78], qui est toujours très utilisé aujourd'hui. Dans cette thèse, nous nous intéresserons à une troisième catégorie de problèmes mathématiques pouvant servir de base à des protocoles cryptographiques, en lien avec les réseaux euclidiens.

Réseaux euclidiens. Un réseau (euclidien) L est un sous-ensemble de $\mathbb{R}^m$ formé de toutes les combinaisons linéaires entières d'un ensemble de n vecteurs linéairement indépendants $b_1, \ldots, b_n \in \mathbb{R}^m$. L'ensemble $(b_1, \ldots, b_n)$ est appelé une base du réseau L, et l'entier n est son rang (cf figure 0.1).

Le problème algorithmique essentiel lié aux réseaux euclidiens est, étant donnée une base arbitraire d'un réseau arbitraire, de trouver un vecteur non nul le plus court du réseau (en norme euclidienne).

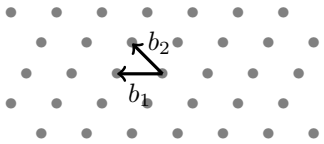

Figure 0.1: Un réseau de dimension 2, avec une base

Ce problème s'appelle le problème du plus court vecteur, abrégé en SVP, pour 'shortest vector problem' en anglais (cf figure 0.2). Asymptotiquement, les algorithmes les plus rapides connus pour résoudre le problème du plus court vecteur sont des algorithmes de crible, dont le temps de calcul est exponentiel en la dimension n du réseau [AKS01, AS18]. Il existe également des variantes de ce problème, qui demandent de trouver un vecteur non nul du réseau dont la norme euclidienne est au plus γ fois la norme d'un vecteur non nul le plus court. Le facteur d'approximation $\gamma \geq 1$ est un paramètre de ce nouveau problème, appelé γ-approx-SVP (cf figure 0.3).

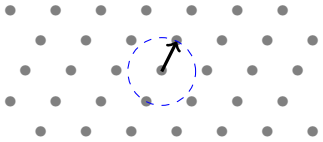

Figure 0.2: Un vecteur non nul le plus court

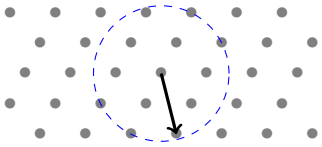

Figure 0.3: Une solution au problème γ-approx-SVP pour $\gamma = 2$

La difficulté de ce problème diminue lorsque γ augmente (le cas $\gamma = 1$ est le problème SVP). En 1987, Schnorr introduisit une hiérarchie d'algorithmes pour résoudre le problème γ-approx-SVP [Sch87]. Les compromis entre temps et facteur d'approximation atteints par les algorithmes de Schnorr sont donnés sur la figure 0.4. En pratique, on utilise l'algorithme BKZ [SE94], qui est une variante heuristique des algorithmes de Schnorr. Une analyse du temps de calcul de l'algorithme BKZ a été effectuée par Hanrot, Pujol et Stehlé [HPS11], montrant que cet algorithme atteint également les compromis temps/facteur d'approximation de la figure 0.4. Le plus petit facteur d'approximation γ pour lequel l'algorithme BKZ termine en temps polynomial est $2^{O(\frac{n \log\log n}{\log n})}$. Il n'existe actuellement pas d'algorithme polynomial résolvant le problème approx-SVP pour des facteurs d'approximation plus petits.

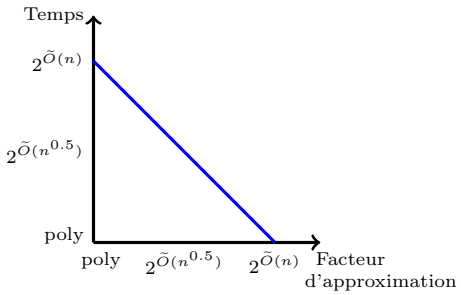

Figure 0.4: Compromis entre temps et facteur d'approximation pour l'algorithme BKZ

Un autre problème important lié aux réseaux euclidiens est le problème du plus proche vecteur (ou CVP, pour 'closest vector problem' en anglais). Étant donnés une base arbitraire d'un réseau arbitraire L, et un point $t \in \mathbb{R}^m$, l'objectif de ce problème est de trouver un point de L le plus proche de t. Encore une fois, il existe une variante de ce problème, appelée γ-approx-CVP, qui consiste à trouver un point de L à distance au plus $\gamma \cdot \text{dist}(L, t)$ de t, où $\text{dist}(L, t)$ est la distance minimale entre t et un point de L (cf figures 0.5 et 0.6). Les meilleurs algorithmes connus pour résoudre γ-approx-CVP ont la même complexité asymptotique que les meilleurs algorithmes pour γ-approx-SVP. En particulier, le meilleur algorithme résolvant CVP exactement a une complexité exponentielle en n et le plus petit

facteur d'approximation qui peut être atteint par un algorithme polynomial est $2^{O(\frac{n \log\log n}{\log n})}$.

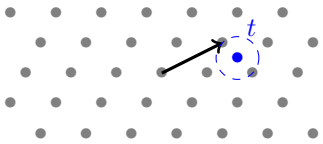

Figure 0.5: Une solution au problème CVP avec cible t

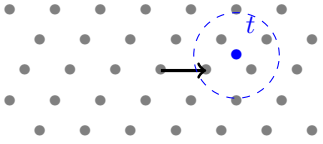

Figure 0.6: Une solution au problème γ-approx-CVP avec cible t et $\gamma = 2$

Cryptographie basée sur les réseaux. Les problèmes SVP et CVP intéressent les cryptographes car ils sont supposés être difficiles à résoudre, même en présence d'un ordinateur quantique. Ils diffèrent en cela des deux autres problèmes de théorie des nombres mentionnés ci-dessus (la factorisation et le logarithme discret), qui peuvent être résolus en temps polynomial quantique grâce à l'algorithme de Shor [Sho97]. Les problèmes de réseaux sont donc de bons candidats pour construire des primitives cryptographiques post-quantique.

La description des problèmes SVP et CVP n'est cependant pas très adaptée à la construction de primitives cryptographiques. La principale raison à cela est la définition de ces problèmes en termes de réseaux arbitraires, alors que les protocoles cryptographiques requièrent en général que le problème soit difficile presque sûrement pour un choix de réseau aléatoire. Cette difficulté a été résolue par l'introduction de nouveaux problèmes, appelés LWE et SIS [Reg05, Ajt96, MR07]. Ces problèmes sont des problèmes cas-moyen, c'est-à-dire qu'il sont supposés être durs à résoudre pour une instance tirée au hasard, ce qui les rend adaptés à la construction de primitives cryptographiques. De plus, il est prouvé que ces problèmes sont au moins aussi difficiles à résoudre que des problèmes pire-cas sur des réseaux, comme par exemple le problème d'approximation de vecteurs courts indépendants (ou approx-SIVP, pour 'approximate short independent vectors problem' en anglais). Étant donnée une base arbitraire d'un réseau arbitraire L, ce problème consiste à trouver n vecteurs de L linéairement indépendants et de norme au plus $\gamma \cdot \lambda_n$, où λ_n est le plus petit réel tel qu'il existe n vecteurs linéairement indépendants dans L plus petits que λ_n. Si l'on suppose qu'une telle variante du problème approx-SVP est difficile à résoudre avec un ordinateur quantique, alors les problèmes LWE et SIS sont également difficiles à résoudre, même avec un ordinateur quantique. On peut donc les utiliser pour construire des primitives cryptographiques post-quantique. Mentionnons également le fait les problèmes LWE et SIS ne sont pas strictement plus difficiles à résoudre que approx-SVP (ou approx-SIVP). En effet, il existe également une réduction montrant que si approx-SVP ou approx-SIVP sont faciles à résoudre, alors SIS et LWE le sont aussi. Les problèmes SIS et LWE sont d'une certaine façon une reformulation du problème SIVP, qui le rend plus adapté à la construction de primitives cryptographiques.

Un exemple de primitive cryptographique pouvant être construite à partir du l'hypothèse LWE est le schéma de chiffrement à clé publique de Regev [Reg05]. Ce schéma de chiffrement est prouvé être sûr sous l'hypothèse que le problème LWE est difficile à résoudre. Depuis 2005, de nombreuses autres primitives cryptographiques reposant sur la supposée difficulté des problèmes LWE et SIS ont été construites. En particulier, les hypothèses à base de réseaux ont permis la construction de primitives avancées, telles que le chiffrement totalement homomorphe.[1] En effet, la première construction de chiffrement totalement homomorphe fut proposée par Gentry en 2009 [Gen09] et utilisaient des hypothèses liées à des réseaux (ainsi qu'une hypothèse de sécurité circulaire). Les hypothèses utilisées pour cette constructions ne sont ni LWE ni SIS, mais une autre construction fut proposée plus tard reposant sur le problème LWE [BV11] (et toujours une hypothèse de sécurité circulaire). Une autre primitive construite en utilisant des techniques liées aux réseaux est l'obfuscation de programmes (qui sera définie dans quelques paragraphes). Bien qu'utilisant des réseaux, les constructions d'obfuscateurs que nous avons aujourd'hui ne sont cependant pas prouvées sûres sous des hypothèses de sécurité standard. Nous reviendrons aux obfuscateurs à la fin de cette introduction.

[1] Un schéma de chiffrement totalement homomorphe est un schéma de chiffrement où l'ont peut effectuer un nombre arbitraire de multiplications et d'additions sur les chiffrés. Cela doit produire un message chiffré qui se déchiffrera en un message correspondant au résultat des même opérations (multiplications/additions), appliquées aux messages sous-jacents.

En conclusion, les problèmes de réseaux tels que LWE et SIS permettent à la fois de construire des primitives (supposées) post-quantique, mais également de concevoir des primitives avancées.

Réseaux structurés. Les protocoles basés sur les réseaux sont généralement moins efficaces que leurs équivalents basés sur la factorisation ou le logarithme discret. La principale raison à cela est qu'un réseau est représenté par une matrice, dont la taille est quadratique en la dimension du réseau. De même, le temps nécessaire pour calculer un produit matrice-vecteur (qui est l'opération de base sur les réseaux) est quadratique en la dimension. Cependant, le temps nécessaire pour résoudre les problèmes SVP ou CVP dans un réseau est seulement exponentiel en la dimension (et non en le carré de la dimension). Ainsi, pour atteindre un niveau de sécurité de λ bits (i.e., la meilleure attaque doit prendre un temps au moins 2^λ), il est nécessaire de travailler avec des paramètres de taille $\Omega(\lambda^2)$.

Afin d'améliorer l'efficacité des protocoles, il est possible de travailler avec des réseaux structurés, comme par exemple des réseaux idéaux. Dans un réseau idéal, les vecteurs correspondent à des polynômes, qui vivent modulo un polynôme P de degré n (de sorte que les polynômes ont tous un représentant de degré $n-1$, qui peut être vu comme un vecteur de dimension n en regardant ses coefficients). Une base d'un réseau idéal correspond à la matrice de multiplication par un polynôme a modulo P (ce qui est bien une opération linéaire).[2] Grâce à cette correspondance avec les polynômes, les matrices peuvent maintenant être représentées par un polynôme (de taille linéaire), et les multiplications matrice-vecteur peuvent être effectuées en temps quasi-linéaire en n (car la multiplication de deux polynômes peut s'effectuer en temps quasi-linéaires en leur degré). De façon plus générale, dans les réseaux idéaux, les vecteurs sont remplacés par des éléments d'un corps de nombre K de dimension n (qui sera $K = \mathbb{Q}[X]/P(X)$ pour le polynôme P défini ci-dessus), et les réseaux sont des idéaux dans ce corps de nombres (d'où la terminologie "réseaux idéaux").

Comme précédemment, il existe des problèmes cas-moyen, appelés Ring-LWE et Ring-SIS [SSTX09, LPR10] (ou RLWE et RSIS). Ces problèmes sont des variantes structurées des problèmes originaux LWE et SIS où, encore une fois, les vecteurs sont remplacés par des polynômes et les matrices correspondent à l'opération de multiplication par un polynôme. Il a été montré que ces problèmes ne sont pas plus faciles à résoudre que le problème approx-SIVP, restreint aux réseaux idéaux. Puisque l'on se restreint maintenant à des réseaux avec une structure algébrique supplémentaire, il est naturel de se demander

Le problème SVP est-il toujours difficile lorsqu'on se restreint à des réseaux idéaux ?

Cette question sera centrale dans cette thèse. Il est également possible d'obtenir des compromis entre efficacité et sécurité, en utilisant des réseaux modules, et leurs problèmes cas-moyen associés module-SIS et module-LWE [BGV14, LS15]. Un réseau module est un réseau dont une base peut être représentée par une matrice par blocs, où chaque bloc est la base d'un réseau idéal.[3] La figure 0.7 représente une base d'un réseau module avec $m \times m$ blocs. La notation B_a désigne une base de l'idéal engendré par a, c'est-à-dire la matrice de multiplication par a modulo P. L'entier m est appelé le rang du module. En variant la taille des blocs, on peut aller des réseaux idéaux (un seul bloc de la taille de la matrice) aux réseaux généraux ($n \times n$ blocs de taille 1). Comme précédemment, il a été prouvé que les problèmes module-SIS et module-LWE sont au moins aussi difficiles que le problème approx-SIVP restreint aux réseaux modules.

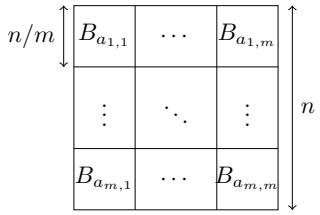

Figure 0.7: Base d'un réseau module de rang m

Faisons quelques remarques sur les réductions inverses. Pour rappel, dans le cas des réseaux généraux, on sait que les problèmes SIS et LWE sont équivalents au problème approx-SIVP. C'est

[2] Cela définit en fait la classe plus restreinte des réseaux idéaux principaux. Afin de simplifier les descriptions dans cette introduction, nous supposerons que tous les idéaux sont principaux.

[3] Encore une fois, on suppose ici que tous les idéaux sont principaux.

également vrai pour les réseaux modules : les problèmes module-LWE et module-SIS sont équivalents au problème approx-SIVP restreint aux modules. En revanche, ce n'est pas le cas des réseaux idéaux. Pour ces réseaux, nous savons simplement que RLWE et RSIS sont au moins aussi difficiles que approx-SIVP dans des réseaux idéaux, mais il se pourrait qu'ils soient strictement plus difficiles. Le problème RLWE est en fait équivalent à approx-SIVP dans des réseaux modules [AD17]. Il se pourrait donc que le problème approx-SIVP pour les réseaux idéaux soit strictement plus facile que le problème RLWE, lui même équivalent à module-LWE et approx-SIVP dans des réseaux modules, eux-mêmes strictement plus faciles que LWE et approx-SIVP.

Concluons ce paragraphe en mentionnant un autre problème de réseau standard : le problème NTRU [HPS98]. Étant donnés un polynôme P définissant un anneau $R = \mathbb{Z}[X]/(P)$, un module $q \in \mathbb{Z}$ et un élément $h = f/g \bmod q$, avec f, g des éléments de R dont les coefficients sont beaucoup plus petits que q, le problème NTRU consiste à retrouver f et g. Contrairement aux problèmes RLWE et RSIS, on ne sais pas si le problème NTRU est plus difficile que certains problèmes pire-cas sur des réseaux structurés. Cependant, ce problème a été étudié pendant plus de 20 ans et est maintenant considéré comme un problème de réseaux standard.

La fonction GGH13. Un schéma de chiffrement par niveaux est une primitive cryptographique permettant à une autorité d'encoder des éléments d'un anneau. Étant donné des éléments encodés, n'importe qui doit ensuite pouvoir publiquement les multiplier ou les additionner, pour obtenir un encodage de l'addition ou de la multiplication des éléments correspondants. Il doit également être possible, après avoir effectué un nombre κ de multiplications (où κ est un paramètre du schéma de chiffrement par niveaux), de tester publiquement si un élément ainsi créé encode zéro ou non. À part cette procédure, appelée test de nullité, les encodages ne doivent rien révéler des éléments encodés. Les schémas de chiffrement par niveaux, s'ils existent, auraient de nombreuses applications en cryptographie. La plus simple, mais que l'on ne sait actuellement pas réaliser à partir d'hypothèses standards, et l'échange de clé en un tour entre $\kappa + 1$ participants, pour un entier $\kappa \geq 3$. Dans un tel protocole, les utilisateurs commencent par générer une paire de clés publique et secrète. Ils diffusent ensuite publiquement leur clé publique à tous les autres utilisateurs. Chacun crée ensuite une clé secrète symétrique s en combinant les clés publiques des autres utilisateurs avec sa propre clé secrète. Cette clé secrète symétrique s doit être la même pour tous les utilisateurs, afin qu'ils puissent ensuite l'utiliser pour communiquer via un protocole symétrique. À l'inverse, un attaquant qui aurait intercepté toutes les clés publiques ne doit pas obtenir d'information sur la clé secrète symétrique s.

La fonction GGH13, proposée par Garg, Gentry et Halevi en 2013 [GGH13a] est le premier candidat de schéma de chiffrement par niveaux. Cette construction utilise des polynômes modulo un polynôme P, et repose sur des techniques similaires à celles utilisées pour les problèmes de réseaux idéaux. Par exemple, les encodages sont de la forme $f/g \bmod q$, pour des polynômes f et g avec de petits coefficients (comme dans le problème NTRU). Ainsi, il est au moins aussi facile d'attaquer la fonction GGH13 que le problème NTRU. De même, la fonction GGH13 utilise des réseaux idéaux, et trouver un vecteur court dans ces réseaux idéaux peut permettre de l'attaquer.

Pour conclure, attaquer la fonction GGH13 est au moins aussi facile que de résoudre certains problèmes standards de réseaux idéaux, mais nous ne connaissons pas de réduction inverse. C'est pourquoi la fonction GGH13 est appelée un candidat : nous ne savons pas comment prouver sa sécurité à partir d'hypothèses standards. En fait, depuis son introduction en 2013, de nombreuses attaques ont été proposées contre la fonction GGH13 et ses applications [HJ16, MSZ16]. Ces attaques ne cassent pas complètement la fonction GGH13, mais elles éliminent un nombre important d'applications possibles du schéma de chiffrement par niveaux.

La principale application de la fonction GGH13 qui n'est pas encore complètement cassée est l'obfuscation. Un obfuscateur est un algorithme qui prend en entrée le code d'un programme et le rend inintelligible, tout en préservant la fonctionalité du programme. En d'autre termes, le nouveau programme doit se comporter comme une boîte noire, il doit permettre à un utilisateur de l'exécuter mais ne doit rien révéler d'autre que le comportement entrée/sortie du programme. Les obfuscateurs sont très désirés en cryptographie et des candidats ont été proposés, utilisant la fonction GGH13 [GGH$^+$13b, GMM$^+$16].[4] Encore une fois, ces constructions sont des candidats, leur sécurité

[4]Tous les candidats obfuscateurs n'utilisent pas la fonction GGH13, mais certains l'utilisent, dont notamment le premier candidat [GGH$^+$13b].

n'est pas prouvée sous des hypothèses de sécurité standards, et un certain nombre de constructions sont sujettes à des attaques [MSZ16, CGH17, CHKL18, Pel18].

Contributions

Pendant ma thèse, je me suis intéressée à la difficulté des problèmes de réseaux structurés. Je me suis concentrée à la fois sur des problèmes fondamentaux, tels que approx-SVP dans des réseaux idéaux ou modules, ainsi qu'à des questions plus concrètes, comme la sécurité de la fonction GGH13 et de certains obfuscateurs l'utilisant.

Approx-SVP dans les réseaux idéaux

Dans le premier chapitre de cette thèse, on s'intéresse à la question déjà mentionnée ci-dessus : le problème (approx-)SVP est-il plus facile lorsque l'on se restreint à des réseaux idéaux ? Une première réponse à cette question a été donnée par Cramer, Ducas et Wesolowski en 2017 [CDW17]. Ils ont montré qu'avec un ordinateur quantique, un attaquant pouvait résoudre en temps polynomial le problème γ-approx-SVP pour un facteur d'approximation γ aussi petit que $\gamma = 2^{\widetilde{O}(\sqrt{n})}$. Cet algorithme ne peut s'utiliser que pour certains polynômes P, définissant des corps cyclotomiques de conducteur une puissance d'un nombre premier. Ces polynômes P sont ceux qui sont les plus utilisés par les constructions cryptographiques. Pour rappel, dans le cas général, le plus petit facteur d'approximation atteignable en temps polynomial (même avec un ordinateur quantique) est $\gamma = 2^{\Omega(n \log\log n / \log n)}$. Ainsi, l'algorithme CDW fournit une accélération quantique par rapport aux réseaux génériques (cf figure 0.8, l'algorithme CDW est responsable du "saut" de la courbe quantique en $\gamma = 2^{\widetilde{O}(\sqrt{n})}$).

Dans le chapitre 3, nous proposons une extension de l'algorithme CDW. Après un pré-calcul exponentiel, notre algorithme atteint tous les compromis intermédiaires entre l'algorithme CDW et l'algorithme BKZ pour γ polynomial. Notre algorithme fournit également une amélioration dans le cas classique, pour de petits facteurs d'approximation (cf figure 0.9). Notre algorithme fonctionne pour n'importe quel polynôme de définition P, mais les compromis de la figure 0.9 dépendent du polynôme P. Le principal inconvénient de notre algorithme est le pré-calcul exponentiel (qui s'effectue en temps $2^{\widetilde{O}(n)}$ pour les corps cyclotomiques). Cela rend notre algorithme inutilisable en pratique. Il convient d'observer cependant que ce pré-calcul ne dépend que du corps de nombres (c'est-à-dire du polynôme P), et peut donc être réutilisé pour différents idéaux dans le même corps de nombre. Cette contribution correspond à la publication suivante.

[PHS19] Alice Pellet-Mary, Guillaume Hanrot, et Damien Stehlé. Approx-SVP in ideal lattices with pre-processing. Dans *Advances in Cryptology – EUROCRYPT*, pages 685–716. Springer, 2019.

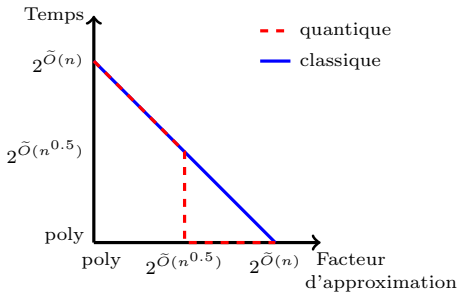

Figure 0.8: Précédents compromis temps/facteur d'approximation pour approx-SVP dans des réseaux idéaux de corps cyclotomiques puissance d'un nombre premier.

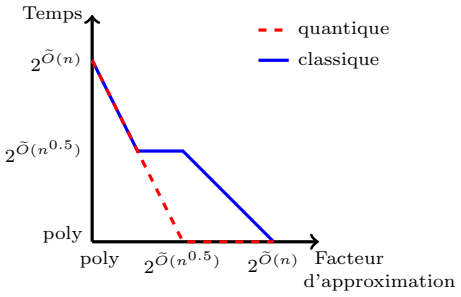

Figure 0.9: Nouveaux compromis pour approx-SVP dans des réseaux idéaux dans les même corps (avec un pré-calcul en temps $\exp(\widetilde{O}(n))$).

Une limitation importante de notre algorithme est qu'il résout le problème approx-SVP seulement dans des réseaux idéaux. Cependant, comme on l'a vu précédemment, on sait que les problèmes

RLWE et RSIS sont au moins aussi difficiles à résoudre que le problème approx-SVP dans des réseaux idéaux, mais la réciproque n'est peut-être pas vraie. Rappelons aussi que la plupart des schéma cryptographiques utilisent les problèmes RLWE et RSIS comme hypothèses de sécurité, et non le problème approx-SVP dans des réseaux idéaux. Ainsi, même s'il existait un algorithme polynomial classique résolvant le problème SVP dans des réseaux idéaux, cela n'aurait que peu d'impact sur la sécurité de la plupart des schémas cryptographiques. Dans les chapitres qui suivent, nous nous intéressons à cette limitation et explorons deux directions. La première direction consiste à essayer d'étendre notre algorithme pour les réseaux idéaux aux réseaux modules. Rappelons que le problème RLWE est équivalent au problème SIVP dans les réseaux modules. Ainsi, trouver des vecteurs courts dans des modules (même de petit rangs 2 ou 3 par exemple), aurait un impact sur les schémas cryptographiques dont la sécurité repose sur les problèmes NTRU et LWE. La seconde direction consiste à s'intéresser à la sécurité de la fonction GGH13, qui est actuellement la principale construction cryptographique dont la sécurité peut être impactée par des algorithmes trouvant des vecteurs courts dans des réseaux idéaux.

Un algorithme LLL pour les modules

Lorsque l'on veut trouver des vecteurs courts dans un réseau module, une idée naturelle est d'essayer de généraliser l'algorithme LLL [LLL82] à l'anneau des entiers d'un corps de nombre. En effet, les blocs dans la base d'un réseau module peuvent être vus comme des éléments d'un anneau R (qui est l'anneau des entiers d'un corps de nombre), ce qui permet de voir la base du module comme une matrice à coefficients dans R, de dimension $m \times m$, où m est le rang du module. Si le rang du module est petit, cette matrice à coefficients dans R aura une petite dimension. L'algorithme LLL (sur $\mathbb{Z}$), permet de trouver en temps polynomial une γ-approximation du plus court vecteur pour un facteur d'approximation γ exponentiel en la dimension. Ainsi, si l'on pouvait utiliser l'algorithme LLL sur notre matrice de petite dimension à coefficients dans R, cela nous permettrait d'obtenir un petit vecteur de notre module en temps polynomial. Rappelons qu'on module avec un petit rang est proche d'un réseau idéal, alors qu'un module avec un grand rang est proche d'un réseau générique. Dans ce dernier cas, même un algorithme LLL dans R ne permettrait pas de résoudre approx-SVP avec un petit facteur d'approximation (car la matrice à coefficients dans R a une grande dimension).

Notre objectif est donc d'essayer d'utiliser notre algorithme résolvant approx-SVP dans les réseaux idéaux pour créer un algorithme LLL dans R. La principale difficulté rencontrée lorsque l'on essaye d'étendre l'algorithme LLL à R est la généralisation de la division euclidienne. En effet, l'algorithme LLL sur $\mathbb{Z}$ repose en grande partie sur le fait que, étant donnés deux entiers a et b, il est possible de trouver $r \in \mathbb{Z}$ tel que $|b + ar| < |a|$ (on peut même faire encore mieux et obtenir $|b + ar| \leq |a|/2$). Pour trouver l'entier r, on utilise la division euclidienne sur $\mathbb{Z}$. Cependant, dans un corps de nombres quelconque, il n'existe pas toujours de division euclidienne. Pire encore, dans la plupart des cas il n'existe même pas d'élément $r \in R$ tel que $\|b + ar\| \leq \|a\|$.

Dans le chapitre 4, nous affaiblissons la condition de division euclidienne et autorisons l'élément b a être multiplié par un petit élément. Plus formellement, notre objectif est, étant donnés $a, b \in R$, de trouver $u, v \in R$ tels que

$$\begin{aligned} \|ua + vb\| &< \|a\| \\ \text{et } \|v\| &\leq C, \end{aligned}$$

pour une certaine constante C à définir (indépendante de $\|a\|$ et $\|b\|$). Nous proposons ensuite un algorithme pour résoudre ce problème. Cet algorithme s'exécute en temps polynomial sur un ordinateur quantique, à condition d'avoir accès à un oracle résolvant le problème CVP dans un réseau fixé, dépendant uniquement de l'anneau R. Malheureusement, la dimension de ce réseau est $\widetilde{O}(n^2)$ (pour un corps cyclotomique), où n est la dimension de l'anneau R. Cela rend l'algorithme inutilisable en pratique. Nous prouvons également que notre affaiblissement de la division euclidienne est suffisante pour étendre l'algorithme LLL aux modules de rang 2 sur R. Cela nous permet ainsi de calculer des vecteurs courts dans des modules de rang 2. Finalement, nous expliquons comment l'algorithme LLL pour les modules de rang 2 peut être étendu à des modules de rang arbitraire. Cela fournit un algorithme LLL sur R, dont la complexité est polynomiale avec un ordinateur quantique, à condition d'avoir accès à un oracle résolvant CVP dans un réseaux fixé, dépendant seulement de R (mais pas du module). Les résultats présentés dans le chapitre 4 correspondent à l'article suivant.

[**LPSW19**] Changmin Lee, Alice Pellet-Mary, Damien Stehlé et Alexandre Wallet. An LLL algorithm for module lattices. Accepté à Asiacrypt 2019.

La fonction GGH13 et ses applications

Dans les deux derniers chapitres de cette thèse, nous nous intéressons à la sécurité de la fonction GGH13 et des obfuscateurs l'utilisant. Nous étudions l'impact des algorithmes résolvant approx-SVP dans des réseaux idéaux sur la fonction GGH13, mais également l'impact d'autres approches, n'utilisant pas la structure algébrique des réseaux apparaissant dans la fonction GGH13.

Fuite statistique. Dans le chapitre 5, nous étudions la sécurité de la fonction GGH13 face aux attaques statistiques. Ces attaques utilisent les propriétés statistiques de la fonction GGH13 et non ses propriétés algébriques, qui avaient été les seules étudiées jusque-là [HJ16, MSZ16]. Notre étude montre que toutes les variantes de la fonction GGH13 sont sujettes à des fuites d'information en lien avec des paramètres secrets de la fonction. La plupart du temps, ces fuites ne semblent pas être suffisantes pour permettre à un attaquant de casser la fonction GGH13, mais pour l'une des variantes (que l'on suspectait déjà d'être sujette à des fuites statistiques), nous avons été capables de transformer ces fuites en une attaque contre la fonction GGH13. Après avoir étudié ces fuites, nous proposons une nouvelle variante de la fonction GGH13, dont nous prouvons qu'elle n'est sujette à aucune fuite d'information dans le modèle considéré dans cette étude. Notons cependant que ce modèle limite grandement les pouvoirs de l'attaquant, et il pourrait exister des modèles moins contraignants où même notre nouvelle variante serait sujette à des fuites statistiques. Ce chapitre correspond à la publication suivante.

[**DP18**] Léo Ducas et Alice Pellet-Mary. On the statistical leak of the GGH13 multilinear map and some variants. Dans *Advances in Cryptology – ASIACRYPT*, pages 465–493. Springer, 2018.

Attaque quantique. Rappelons qu'un schéma de chiffrement par niveaux possède une procédure qui permet à un utilisateur de tester si un encodage encode zéro, après avoir effectué κ multiplications. Les auteurs de la fonction GGH13 [GGH13a] avaient déjà observé que s'il est possible de trouver des vecteurs courts dans certains réseaux idéaux, alors il devient possible d'effectuer un test de nullité après 2κ multiplications au lieu de κ. Pour cela, il n'est même pas nécessaire de savoir trouver des vecteurs courts dans n'importe quel réseau idéal, mais simplement dans ceux possédant un petit générateur. Pour ces réseaux idéaux, il existe un algorithme polynomial quantique permettant de trouver un vecteur le plus court [CDPR16]. Cela signifie qu'il est possible, avec un ordinateur quantique, d'effectuer des tests de nullité illégaux, après 2κ multiplications.

Il est important de remarquer cependant que même s'il est possible d'effectuer des tests de nullité après 2κ multiplications, cela n'implique pas immédiatement qu'il existe des attaques contre les constructions utilisant la fonction GGH13. Dans le chapitre 6, nous nous intéressons à l'impact de ces tests illégaux sur les constructions d'obfuscateurs utilisant la fonction GGH13. Nous montrons que pour un grand nombre de constructions, ces tests illégaux peuvent être utilisés pour créer des attaques contre les obfuscateurs. Cette attaque quantique correspond à la publication suivante.

[**Pel18**] Alice Pellet-Mary. Quantum attacks against indistinguishablility obfuscators proved secure in the weak multilinear map model. Dans *Advances in Cryptology – CRYPTO*, pages 153–183. Springer, 2018.

Une remarque sur les heuristiques

Tous les résultats présentés dans cette thèse reposent sur des heuristiques. Tout au long du manuscrit, nous essayons d'identifier clairement les heuristiques, en les numérotant et les mettant dans des environnements particuliers. Nous précisons également, pour chaque théorème, s'il est heuristique et, si oui, quelles heuristiques sont utilisées. Pour les heuristiques introduites dans cette thèse (par opposition aux heuristiques déjà présentes dans des travaux antérieurs), nous proposons une justification mathématique, ainsi que des expériences numériques les corroborant (lorsque c'est possible). L'introduction de chaque chapitre précise l'adresse internet où est disponible le code utilisé pour effectuer les expériences présentées dans le chapitre.

Chapter 1

Introduction

One of the main objectives of cryptography is to guarantee the confidentiality of messages, by *encrypting* them during the transmission phase. Historically, the encryption procedure relied on a *secret symmetric key*, which was shared between the sender and the receiver and nobody else. Such a scheme was called a *symmetric key* encryption scheme, and required the sender and the receiver to meet in person, or use a trusted third party, before being able to communicate safely. For today's use of cryptography, for example for communications over the Internet, the requirement that the sender should physically meet the receiver before any communication is not reasonable. Fortunately, Diffie and Hellman proposed in 1976 a first *public key* cryptographic scheme [DH76]. The principle of public key encryption schemes is that each user now has its own secret key, which it shares with nobody else, but it also has a public key, which it can make publicly available. This public key enables anyone to encrypt a message, however, only the user with the corresponding secret key will be able to recover the original message. In other words, given only the public key and an encrypted message, no information should be leaked about the corresponding plaintext message. A public key encryption scheme can be used to preserve confidentiality of messages without the need for a physical meeting beforehand. Indeed, the receiver can send its public key to the sender. The sender then encrypts the message using the public key and sends it back to the receiver. An eavesdropper listening to the conversation will only know the public key and the encrypted message, which by construction should not reveal anything about the message. In today's cryptographic communication protocols, a public key scheme is typically used at the beginning of the conversation to exchange a shared secret symmetric key between two users, and then a symmetric key scheme is used with this shared secret symmetric key for the rest of the conversation. This procedure is more efficient than relying on public key encryption schemes for the full conversation, as the symmetric key schemes are usually more efficient than the public key ones.

The security of public key encryption schemes is typically based on "simple" assumptions, i.e., assumptions which can be described in a few sentences. Namely, under the assumption that some mathematical problems are hard to solve, one can prove that no attacker (for a certain model of attacker) can recover the underlying message by knowing only the encrypted message and the public key. The Diffie-Hellman protocol mentioned above is based on a problem related to the discrete logarithm problem. Given a cyclic group G, a generator of the group g and an arbitrary element $h \in G$, the discrete logarithm problem consists in computing the logarithm of h in base g, i.e., finding an integer α such that $h = g^\alpha$. The presumed intractability of this problem has since been used for many other cryptographic constructions. Another very important number theoretic problem for cryptography is factorization. Given as input the product pq of two large prime numbers p and q, this problem consists in recovering the two primes p and q. A problem related to this one is used to assess the security of the RSA protocol [RSA78], which is still widely used today. In this thesis, we will be interested in a third category of mathematical problems serving as cryptographic hardness assumptions, related to Euclidean lattices.

Lattices. A (Euclidean) lattice L is a subset of $\mathbb{R}^m$ consisting of all integer linear combinations of a set of n linearly independent vectors $b_1, \ldots, b_n \in \mathbb{R}^m$. The set of vectors $(b_1, \ldots, b_n)$ is called a basis of the lattice L, and the integer n is its rank (see Figure 1.1).

The fundamental algorithmic problem related to lattices is, given an arbitrary basis of an arbitrary lattice, to find a shortest non-zero element of the lattice (with respect to its Euclidean norm). This problem is referred to as the shortest vector problem, or SVP for short (see Figure 1.2). Asymptotically,

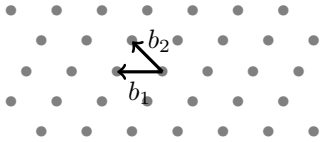

Figure 1.1: A two-dimensional lattice with a basis

the fastest algorithms that are known to solve the shortest vector problem are sieving algorithms, whose running time is exponential in the dimension n of the lattice [AKS01, AS18]. Approximation variants of the shortest vector problem can also be considered, where the problem consists in finding a non-zero vector of the lattice whose Euclidean norm is no more than γ times the norm of a shortest non-zero vector. The approximation factor $\gamma \geq 1$ is a parameter of this new problem, which is referred to as the γ-approximate shortest vector problem, or γ-approx-SVP for short (see Figure 1.3).

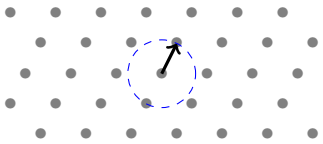

Figure 1.2: A shortest non-zero vector in a lattice

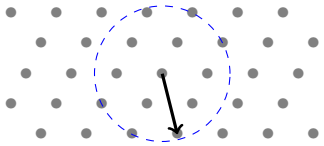

Figure 1.3: A solution to γ-approx-SVP for $\gamma = 2$

The difficulty of this problem decreases when γ increases (observe that 1-approx-SVP is the regular SVP). In 1987, Schnorr introduced a hierarchy of algorithms solving approx-SVP [Sch87], achieving the trade-offs between time and approximation factor drawn in Figure 1.4. In practice, one uses the BKZ algorithm [SE94], which is a heuristic variant of Schnorr's algorithms. A proof of the running time of the BKZ algorithm was provided in [HPS11], showing that the BKZ algorithm also achieves the trade-offs of Figure 1.4. The smallest approximation factor γ for which the BKZ algorithm runs in polynomial time is $2^{O(\frac{n \log\log n}{\log n})}$. We do not know any other polynomial time algorithm solving approx-SVP for asymptotically smaller approximation factors.

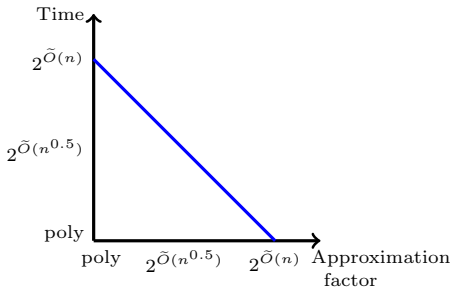

Figure 1.4: Time/approximation trade-offs achieved by BKZ for arbitrary lattices

Another important problem related to lattices is the closest vector problem (or CVP for short), which requires, given an arbitrary target point t in $\mathbb{R}^m$ and an arbitrary basis of an arbitrary lattice L, to find a point of L closest to t. Again, we have a variant of this problem, called γ-approx-CVP, where the requirement is to find a point of L at distance at most $\gamma \cdot \mathrm{dist}(L, t)$ from t, where $\mathrm{dist}(L, t)$ is the minimal distance between t and a point of L, and $\gamma \geq 1$ (see Figures 1.5 and 1.6). The best algorithms to solve γ-approx-CVP are asymptotically as efficient as for γ-approx-SVP. In other words, the best algorithm to solve exact CVP has a complexity exponential in the dimension n of the lattice and the smallest approximation factor for which we have a polynomial time algorithm is $2^{O(\frac{n \log\log n}{\log n})}$.

Lattice-based cryptography. The shortest vector problem and closest vector problem mentioned above are interesting for cryptographic purposes, as they are conjectured to be hard to solve even in

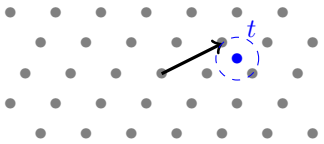

Figure 1.5: A CVP instance with target t

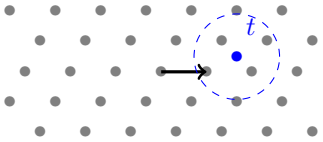

Figure 1.6: A γ-approx-CVP instance with target t for $\gamma = 2$

the quantum setting. To our current knowledge, the existence of a quantum computer would only decrease the constant in the exponent of the best algorithm solving these problems, but they would remain exponentially hard. This is a major difference with the two other number theoretic problems mentioned above (i.e., the discrete logarithm and the factorization problems). Indeed, Shor described in 1994 a quantum algorithm which can be used to solve these two problems in polynomial quantum time [Sho97]. Lattice problems are then good candidates to construct post-quantum cryptography.

The description of the problems as CVP or SVP is however not very suited for constructing cryptographic primitives. The main reason is their definition in terms of arbitrary lattices, whereas cryptographic design typically requires almost-always hardness for instances drawn randomly from specific distributions. This difficulty was circumvented by the introduction of new lattice problems, namely the LWE and SIS problems [Reg05, Ajt96, MR07]. These problems are average case problems, meaning that they are supposed to be hard for a randomly chosen instance, which makes them more suitable for building cryptographic primitives. Moreover, these problems have been shown to be no easier to solve that some worst case lattices problems, such as the approximate short independent vectors problem (or approx-SIVP for short). Given as input an arbitrary basis of an arbitrary lattice L, the γ-approx-SIVP problem requires to find n linearly independent vectors of L shorter than $\gamma \cdot \lambda_n$, where λ_n is the smallest real number such that there exists n linearly independent vectors in L of norm bounded by λ_n.[1] Assuming that such a variant of the approximate shortest vector problem is quantumly intractable implies that the LWE and SIS problems are also quantumly intractable and can be used to build post-quantum cryptographic primitives. Let us also mention that the LWE and SIS problems are not strictly harder than approx-SVP (or approx-SIVP). Indeed, there also exists a reduction showing that if approx-SVP (or approx-SIVP) is easy to solve, then so are LWE and SIS. These problems are in a sense only average-case reformulations of approx-SIVP, which makes it more suitable for constructing cryptographic primitives.

One example of a cryptographic primitive which can be constructed from the LWE assumption is Regev's public key encryption scheme [Reg05]. This encryption scheme is proven to be secure, under the assumption that the LWE problem is hard to solve. Since then, many other cryptographic primitives have been constructed, based on the supposed hardness of LWE or SIS. In particular, lattice assumptions enabled the construction of advanced cryptographic primitives, such as fully homomorphic encryption.[2] Indeed, the first construction of a fully homomorphic encryption scheme was proposed by Gentry in 2009 [Gen09] and relied on lattice assumptions (as well as a circular security assumption). The lattice assumptions used in this construction differ from the LWE or SIS assumptions, but another construction was proposed [BV11], based on the plain LWE problem (and a similar circular security assumption). Another primitive which has been constructed using lattice techniques is program obfuscation (which will be defined in a few paragraphs). However, even if they use lattices, the constructions of obfuscators we currently have are not proved secure under standard lattice assumptions. We will come back to obfuscation at the end of this introduction.

To sum up, lattice problems such as LWE or SIS enable us to both construct (supposedly) post-quantum primitives, and to design advanced primitives.

[1] Observe that while this problem might seem harder to solve than approx-SVP, it is in fact not the case. Indeed, even if we can find n vectors smaller than $\gamma \cdot \lambda_n$, it could be the case that none of them is shorter than $\gamma \cdot \lambda_1$, where λ_1 is the norm of a shortest non-zero vector of the lattice.

[2] A fully homomorphic encryption scheme is an encryption scheme which is additively and multiplicatively homomorphic for an arbitrary number of additions and multiplications. In other words, (an arbitrary number of) additions/multiplications of the encrypted messages should produce ciphertexts that decrypt to the additions/multiplications of the underlying messages.

Structured lattices. Protocols based on lattice assumptions are usually less efficient (in terms of public key size and ciphertexts size) than their counterparts based on the discrete logarithm problem or on factorization. This is mainly due to the fact that a lattice is represented by a matrix, whose size is quadratic in the dimension of the lattice. Similarly, the time needed to compute a matrix-vector multiplication (which is the basic operation on lattices) is quadratic in the dimension. The difficulty of solving SVP or CVP in a lattice however only increases exponentially in the dimension (and not in the square of the dimension). This implies that to achieve a given level of security λ (meaning that the best attack should run in time at least 2^λ), one has to work with parameters of size $\Omega(\lambda^2)$.

In order to improve efficiency, one may want to work with structured lattices. One such example of structured lattices are ideal lattices. In an ideal lattice, the vectors correspond to polynomials, which live modulo some fixed irreducible polynomial P of degree n (hence, each polynomial has n coefficients and can be mapped to a vector of dimension n, by looking at its coefficients). An ideal lattice then corresponds to multiplication by some polynomial a (multiplication by a polynomial a modulo P is a linear operation which can be represented by a matrix).[3] Thanks to this correspondence with polynomials and polynomial multiplication, the matrices can now be stored with only n elements and all operations are quasi-linear in n (the multiplication of two polynomials can be done in time quasi-linear in their degree). A more algebraic way to say the same thing is that the vectors are replaced by elements of a number field of dimension n (which will be $K = \mathbb{Q}[X]/P(X)$ for the polynomial P defined above), and the lattices are ideals in this number field. This is where the terminology "ideal lattice" comes from.

As previously, some average-case problems can be defined, called Ring-LWE [SSTX09, LPR10] and Ring-SIS [LM06, PR06] (or RLWE and RSIS for short). These problems are structured variants of the original LWE and SIS problems, where, again, the vectors are seen as polynomials and the matrices as polynomial multiplication. These problems are known to be no easier to solve than solving approx-SIVP restricted to *ideal* lattices (which, for the specific case of ideal lattices, is equivalent to approx-SVP). Observe that we are now considering the shortest vector problem in a restricted class of lattices, with an extra algebraic structure. One may then wonder

Is SVP still hard when restricted to ideal lattices?

This will be a central question of this thesis. One can also obtain trade-offs between efficiency and security by using module-lattices, and their average case problems Module-SIS and Module-LWE [BGV14, LS15]. A module lattice is a lattice whose basis is a block matrix, each block being the basis of an ideal lattice.[4] Figure 1.7 represents the basis of a module lattice, with m^2 blocks of size $n/m \times n/m$ each (the dimension n of the lattice has to be a multiple of m). The notation B_a refers to a basis of the ideal generated by a, i.e., it is the matrix representing multiplication by a modulo P. The integer m is called the rank of the module. By varying the size of these blocks, one can go from ideal lattices (one big block of the size of the matrix) to general lattices (n by n blocks of dimension 1×1). As in the previous cases, the Module-SIS and Module-LWE problems are proven to be no easier to solve than the approx-SIVP problem restricted to module lattices.

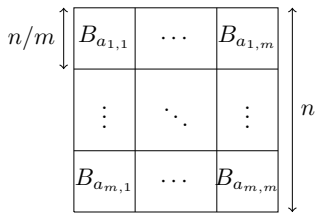

Figure 1.7: Basis of a module lattice of rank m

Let us make some remarks on the converse reductions. Recall that in general lattices, the LWE and SIS problems are equivalent to the approx-SIVP problem (i.e., we have reductions in both directions).

[3] This in fact defines the more restricted class of *principal* ideal lattices. For simplicity, in this introduction we are going to pretend that all ideals are principal.

[4] Once again, we assume for simplicity in this introduction that all ideals are principal (see Section 2.3.7 for a definition of module lattices and pseudo-bases in the general case).

The same holds also for Module-SIS/LWE and approx-SIVP in modules lattices. However, in the case of ideal lattices, we only have one direction. It could be that approx-SVP in ideal lattices is in fact strictly weaker than RLWE and RSIS. The RLWE problem is in fact known to be equivalent to approx-SIVP in module lattices [AD17]. Hence, it could be that approx-SVP in ideal lattices is strictly weaker than RLWE, which is equivalent to Module-LWE and Module-SIVP, which is itself strictly weaker than LWE and approx-SIVP.

Finally, let us conclude this paragraph by mentioning another standard lattice assumption based on structured lattices: the NTRU problem [HPS98]. Given as input a polynomial P defining the ring $R = \mathbb{Z}[X]/(P)$, a modulus $q \in \mathbb{Z}$ and an element $h = f \cdot g^{-1} \bmod q$, with f, g elements of R with coefficients much smaller than q, the NTRU problem asks to recover the elements f and g. Unlike the previous RLWE and RSIS problems, the NTRU problem is not known to be harder than some worst case problem in structured lattices. However, it has been studied for more than 20 years and is now considered as a standard lattice problem.

The GGH13 map. A graded encoding scheme is a primitive which allows an authority to encode elements of a ring. Anyone should then be able to publicly add or multiply the produced encodings, to obtain encodings of the addition or multiplication of the corresponding encoded elements. One should also be able, after having performed a fixed number κ of multiplications (where κ is a parameter of the scheme), to test whether the created encoding is an encoding of zero or not. Except for this zero-test, no other information should be leaked about the elements being encoded. Graded encoding schemes, if they exist, would have a lot of applications in cryptography. One of the simplest applications, but which we currently do not know how to instantiate from standard assumptions, is one-round key exchange between $\kappa + 1$ users, for some integer $\kappa \geq 3$. In a one-round key-exchange protocol, all the $\kappa + 1$ users first generate a pair of public and secret keys. In a second phase, they publicly broadcast their public key to all other users. Finally, they recover the public keys of the other users and combine them with their own secret key to create a secret string s. What is important is that all users should create the same secret string s, which can then be used to communicate using a symmetric key encryption scheme. On the other hand, an attacker which intercepts all the public keys should not be able to learn anything about the secret s.

The GGH13 map, proposed by Garg, Gentry and Halevi in 2013 [GGH13a] was the first candidate graded encoding scheme. The construction uses polynomials modulo some fixed polynomial P, and is based on techniques similar to the structured lattices ones. For instance, the encodings are NTRU-like elements (i.e., they are of the form $f/g \bmod q$ for some polynomials f and g with small coefficients), hence the GGH13 map is at most as hard as the NTRU problem. There are also ideals involved in the construction of the GGH13 map, and one has to assume that finding short vectors of these ideals is a hard problem for the security of the map.

Overall, one has to assume the difficulty of several standard lattice problems for the GGH13 map to be secure, but we do not have any converse reduction. This is why the GGH13 map is called a candidate graded encoding scheme: we do not know how to prove its security based on standard assumptions. In fact, since its introduction in 2013, multiple attacks have been proposed against the GGH13 map and its applications [HJ16, MSZ16]. These attacks do not fully break the GGH13 map, but they rule out many possible applications of the graded encoding scheme.

The main application of the GGH13 map which is not fully broken yet is obfuscation. An obfuscator is an algorithm which should take as input the code of a program, and output a new code, which computes the same function as the original code, but should not reveal anything about it. In other words, it should act as a black-box, enabling a user to evaluate the code, without revealing anything about it, except its input/output behavior. Obfuscation is a much desired cryptographic primitive, and some candidate obfuscators have been proposed, using the GGH13 map [GGH$^+$13b, GMM$^+$16].[5] Again, these constructions are only candidates, their security is not proved under standard assumptions, and many constructions suffer from attacks [MSZ16, CGH17, CHKL18, Pel18].

[5] Since the description of the first candidate obfuscator [GGH$^+$13b] which was based on the GGH13 map, other candidate constructions have been proposed and some of them do not rely on the GGH13 map.

1.1 Contributions

During my PhD, I have been interested in the difficulty of problems related to structured lattices. I have focused on both fundamental problems, such as approx-SVP in ideal lattices or in module lattices, as well as more concrete questions, such as the security of the GGH13 map and some candidate obfuscators using it.

1.1.1 Approx-SVP in ideal lattices

The first chapter of this thesis focuses on the question already mentioned above: is (approx-)SVP easier when restricted to ideal lattices? A first answer to this question was provided by Cramer, Ducas and Wesolowski in 2017 [CDW17]. They showed that in the quantum setting, an attacker can solve in polynomial time the γ-approx-SVP problem in ideal lattices for approximation factors as small as $\gamma = 2^{\widetilde{O}(\sqrt{n})}$. This algorithm only works for certain choices of polynomial P (recall that we are working with polynomials modulo P), corresponding to prime power cyclotomic number fields. These choices of P are the ones that are the most used for cryptographic constructions. Recall that in the case of general lattices, the smallest approximation factors we can achieve in polynomial time (even in the quantum setting) are $\gamma = 2^{\Omega(n \log\log n / \log n)}$. Hence, the CDW algorithm provides a quantum speedup for ideal lattices compared to general lattices (see Figure 1.8, the trade-offs in blue are the ones achieved by the BKZ algorithm, and the CDW algorithm is responsible for the "jump" in the quantum setting at $\gamma = 2^{\widetilde{O}(\sqrt{n})}$).

In Chapter 3, we present an extension of the CDW algorithm. After an exponential pre-processing phase, our algorithm achieves all the intermediate trade-offs in the quantum setting between the CDW algorithm, and the exponential time algorithms for polynomial approximation factors. In addition, our algorithm also improves upon the BKZ trade-offs in the classical setting, for small approximation factors (see Figure 1.9). Finally, our algorithm works in any number field, i.e., for any defining polynomial P (but the trade-offs shown on the figure change with the number fields). The major drawback of our algorithm is the exponential pre-processing phase (which runs in time $2^{\widetilde{O}(n)}$ for prime power cyclotomic fields). This makes our algorithm unusable in practice. We note however that this pre-processing phase only depends on the number field (i.e., on the polynomial P), hence, once it is done, it can be reused for any ideal lattice defined over this number field. This contribution corresponds to the following publication.

[PHS19] Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. Approx-SVP in ideal lattices with pre-processing. In *Advances in Cryptology – EUROCRYPT*, pages 685–716. Springer, 2019.

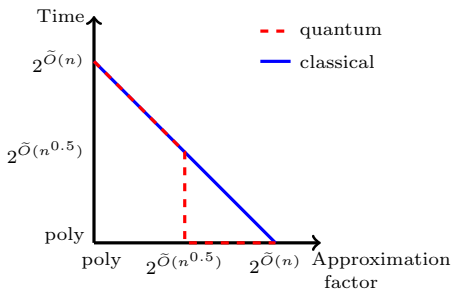

Figure 1.8: Prior time/approximation trade-offs for ideal approx-SVP in prime power cyclotomic fields.

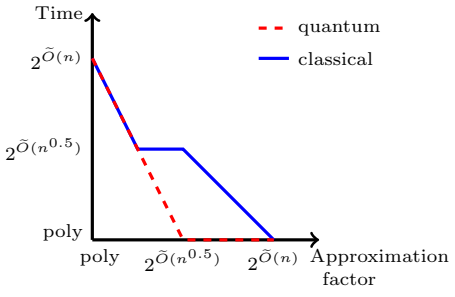

Figure 1.9: New trade-offs for ideal approx-SVP in the same fields (with a pre-processing of cost $\exp(\widetilde{O}(n))$).

A fundamental limitation of this algorithm is that it focuses on finding short vectors in ideal lattices. However, recall that we have seen that the RLWE and RSIS problems are proven to be no easier to solve than approx-SVP in ideal lattices, but the reverse reduction is not known. Recall also that most of the cryptographic schemes are based on the RLWE or RSIS problems, and not on approx-SVP in ideal lattices. Hence, even a classical polynomial time algorithm for finding short vectors in ideal lattices

would have little impact on the security of most of the structured-lattice-based cryptographic schemes. In the following chapters, we will focus on this limitation and explore two main directions. The first direction consists in trying to extend our algorithm from ideal lattices to module lattices. As mentioned above, the RLWE problem is equivalent to SIVP in module lattices. Hence, being able to find short vectors in modules (even modules of very small rank,[6] like for instance 2 or 3) would have a direct effect on cryptographic schemes based on NTRU and RLWE. The second direction consists in focusing on the security of the GGH13 map, which is currently the main construction whose security can be impacted by approx-SVP solvers for ideal lattices.

1.1.2 An LLL algorithm for modules

When trying to extend the SVP solver to module lattices, a natural idea is to try to generalize the LLL algorithm [LLL82] over the ring of integers of a number field. Indeed, the blocks of the basis of a module lattice can been seen as ring elements (in the ring of integers R of a number field), which provides a matrix with coefficient in R instead of $\mathbb{Z}$. If the number of blocks of our module is small, then this matrix has small dimension. The LLL algorithm (over $\mathbb{Z}$) finds in polynomial time a γ-approximation of the shortest vector of the lattice, for an approximation factor γ which is exponential in the dimension. Hence, if we could apply the LLL algorithm to our matrices with coefficients in R and small dimension, we would obtain in polynomial time a small approximation of the shortest non-zero vector of the corresponding module lattices. Recall that modules with a small number of blocks are close to being ideal lattices. On the contrary, if the number of blocks is large, then the modules are close to general lattices, and even an LLL algorithm over R would not find small approximation factors (because the dimension of the basis as a matrix over R would be large).

The question we focus on is then to try to use our approx-SVP solver in ideal lattices to extend the LLL algorithm to R. The main difficulty one faces when trying to extend the LLL algorithm to a ring of integers R, is to generalize the Euclidean division. Indeed, the LLL algorithm over $\mathbb{Z}$ crucially relies on the fact that given two integers $a, b \in \mathbb{Z}$, one can find an element $r \in \mathbb{Z}$ such that $|b + ra| < |a|$ (one can even achieve $|b + ra| \leq |a|/2$). This means that we can always reduce b, with multiples of a, to obtain a new element strictly smaller than a. This reduction is provided over $\mathbb{Z}$ by the Euclidean division. The Euclidean division however does not generalize to all rings of integers. Even more, in most cases, there will not even exist an element $r \in R$ such that $\|b + ar\| \leq \|a\|$.

In Chapter 4, we relax the condition on the Euclidean division by allowing b to be multiplied by a small element. More formally, our objective is, given $a, b \in R$, to find $u, v \in R$ such that

$$\begin{gathered} \|ua + vb\| < \|a\| \\ \text{and } \|v\| \leq C, \end{gathered}$$

for some constant C not too large. We then describe an algorithm which solves this problem. This algorithm runs in quantum polynomial time if it is given access to an oracle solving CVP in a fixed lattice, depending only on the ring R. Unfortunately, this lattice has dimension $\widetilde{O}(n^2)$,[7] where n is the dimension of the ring R, which makes the algorithm not implementable in practice. We also prove in Chapter 4 that this relaxation of the Euclidean division is sufficient to extend the LLL algorithm to rank-2 modules over R. Hence, we obtain an algorithm computing short vectors in modules of rank 2. Finally, we explain how the LLL algorithm for rank-2 modules can be extended to modules of arbitrary rank. This gives us an LLL algorithm over R, which runs in quantum polynomial time if given access to an oracle solving CVP in a fixed lattice depending only on R (and not on the module). Chapter 4 corresponds to the following work.

[LPSW19] Changmin Lee, Alice Pellet-Mary, Damien Stehlé and Alexandre Wallet. An LLL algorithm for module lattices. Accepted at Asiacrypt 2019.

[6] Recall that the rank of a module corresponds to its number of blocks: a module of rank m has a matrix basis formed by $m \times m$ blocks of ideal lattices.

[7] In the case of cyclotomic number fields. For arbitrary number fields, the dimension could even be larger than $\widetilde{O}(n^2)$.

1.1.3 The GGH13 map and its applications

In the last two chapters of this thesis, we focus on the security of the GGH13 map and its application to obfuscation. We have seen that finding short vectors in ideal lattices has an impact on the security of the GGH13 map. We study this impact, but also consider another approach, which does not rely on the structured lattices appearing in the GGH13 map.

Statistical leakage. In Chapter 5, we first start by studying the security of the GGH13 map against statistical attacks. These attacks focus on the statistical properties of the GGH13 map and not on its algebraic properties, which had been the only ones studied so far [HJ16, MSZ16]. Our study shows that all the variants of the GGH13 map leak some information related to some secret parameters of the map. Most of the time, this leakage does not seem to provide a way to attack the GGH13 map, but for one of the variants (which was already suspected to be subject to statistical attacks), we were able to transform the leakage into an attack against the GGH13 map. After studying the leakage of all the variants, we derive a new variant of the GGH13 map which provably does not leak any secret information in the model considered in this study. We note however that the attacker we consider in this work is very restricted and there might exist other statistical attacks which can recover secret information about the GGH13 map, even with our new variant. This chapter corresponds to the following publication.

[**DP18**] Léo Ducas and Alice Pellet-Mary. On the statistical leak of the GGH13 multilinear map and some variants. In *Advances in Cryptology – ASIACRYPT*, pages 465–493. Springer, 2018.

Quantum attack. Recall that a graded encoding scheme has a public procedure which allows a user to zero-test an encoding (i.e., determine whether the encoding is an encoding of zero) after a fixed number κ of multiplications. It was already observed by the authors of the GGH13 map [GGH13a] that being able to find short vectors in ideal lattices enables an attacker to zero-test after an illegal number of multiplications, namely 2κ multiplications. To do so, an attacker does not even need an SVP solver for arbitrary ideal lattices, but only for principal ideals with a small a generator. For this specific class of ideals, the short principal ideal problem solver of Cramer, Ducas, Peikert and Regev [CDPR16] provides a polynomial time quantum algorithm solving SVP. Hence, the GGH13 map was already known to be subject to these illegal zero-tests in the quantum setting.

One important remark however is that being able to zero-test after 2κ multiplications instead of κ does not provide an immediate attack for all the constructions using the GGH13 map. In Chapter 6, we focus on the impact of this illegal zero-test on obfuscators using the GGH13 map. We show that for a large number of them, this illegal zero-test can be used to mount a concrete attack against the obfuscators. This quantum attack corresponds to the following publication.

[**Pel18**] Alice Pellet-Mary. Quantum attacks against indistinguishablility obfuscators proved secure in the weak multilinear map model. In *Advances in Cryptology – CRYPTO*, pages 153–183. Springer, 2018.

1.1.4 A note on heuristic assumptions

All the results presented in this thesis rely on heuristic assumptions. Throughout the thesis, we try to clearly identify all the heuristics which are used, by putting them in a specific environment and numbering them. We also specify in each theorem whether the theorem is heuristic, and if so which heuristics are used. For the heuristics which are introduced for this thesis (as opposed to the heuristics that were already used in prior works), we try to provide a justification based on mathematical arguments as well as numerical experiments (when it is possible). In every chapter, the introduction provides a url where one can find the code used to perform the experiments related to the chapter.

Chapter 2

Preliminaries

In this chapter, we recall some preliminary results. In a first section, we focus on lattices and related algorithmic problems. We then recall some properties and mathematical objects related to number fields. In particular, we define ideals and modules, and explain how they can be seen as lattices. We also define the class group and present some algorithmic problems related to its computation. Finally, we give some definitions related to statistics and discrete Gaussians, and we conclude by recalling the definition of matrix branching programs.

2.1 Notations

We let $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$ denote the sets of integers, rational, real, and complex numbers, respectively. We write $\mathbb{Z}_{\geq 0}$ and $\mathbb{R}_{\geq 0}$ the sets of non-negative integers and non-negative real numbers. For any positive integer n and prime power q, we let $\mathbb{Z}/n\mathbb{Z}$ denote the set of integers modulo n and $\mathbb{F}_q$ denote the finite field of cardinality q. For $x \in \mathbb{C}$, we let $\bar{x}$ denote its complex conjugate. For $x \in \mathbb{R}$, we write $\lfloor x \rceil$ the nearest integer to x.

For a positive real number x, we let $\log x$ denote its binary logarithm and $\ln x$ denote its natural logarithm. For two functions $f(n)$ and $g(n)$, we write $f(n) = \widetilde{O}(g(n))$ if there exists some constant $c > 0$ such that $f(n) = O(g(n) \cdot |\log g(n)|^c)$. We abuse notations by defining $\widetilde{O}(n^\alpha) = O(n^\alpha \operatorname{poly}(\log n))$ even if $\alpha = 0$ (this will simplify some statements). We say that "ε is negligible in n" if $\varepsilon = \varepsilon(n)$ is a function of n which is asymptotically smaller than any inverse polynomial in n, i.e., $\varepsilon(n) = \frac{1}{n^{\omega(1)}}$.

For a vector $v \in \mathbb{R}^n$, we let v_i denote the i-th coordinate of v. We also let v^T denote the transpose of v. Given a finite set X, we let $|X|$ denote the cardinality of X. For any ring $\mathcal{R}$, we write $\mathcal{R}^\times$ the set of invertible elements of $\mathcal{R}$ and $\mathcal{R}^* := \mathcal{R} \setminus \{0\}$.

2.2 Lattices

We start by defining lattices and describing some of their properties. In this thesis, all the vectors are column vectors.

Definition 2.1. A lattice $L \subset \mathbb{R}^s$ is the set of all linear integer combinations of a set $(b_i)_{1 \leq i \leq n}$ of n linearly independent vectors of $\mathbb{R}^n$. In other words,

$$L = \{\sum_{i=1}^{n} a_i b_i \ : \ a_1, \ldots, a_n \in \mathbb{Z}\}.$$

The set of vectors $(b_i)_{1 \leq i \leq n}$ is called a basis of the lattice. The integer n is called the rank of L. The lattice L is said to be full-rank if $n = s$.

The inclusion of a lattice L in the Euclidean space $\mathbb{R}^s$ naturally equips L with different norms. For an element $x = (x_1, \ldots, x_s)^T \in \mathbb{R}^s$, the Euclidean norm (or ℓ_2-norm) is defined by $\|x\|_2 = \sqrt{\sum_i x_i^2}$. This is the norm that is used most of the time in this thesis, and when there is no ambiguity, we simply write it $\|x\|$. The infinity norm (or ℓ_∞-norm) is defined by $\|x\|_\infty = \max_i |x_i|$ and the ℓ_1-norm is defined by $\|x\|_1 = \sum_i |x_i|$. Note that only the ℓ_2-norm is invariant under orthonormal transformations. For $x, y \in \mathbb{R}^s$, we let $\langle x, y \rangle = \sum_i x_i y_i$ denote the usual scalar product (inducing the Euclidean norm). For

any $x \in \mathbb{R}^s$ and $y \in \mathbb{Z}^s$, the following properties holds

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{s} \cdot \|x\|_\infty \tag{2.1}$$

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{s} \cdot \|x\|_2 \tag{2.2}$$

$$\|y\|_1 \leq \|y\|_2^2 \tag{2.3}$$

For a lattice L and $i \in \{1, 2, \infty\}$, we let $\lambda_1^{(i)}(L)$ denote the norm of a shortest non-zero vector of L for the ℓ_i-norm. Similarly, for $k \geq 1$, we let $\lambda_k^{(i)}(L)$ denote the smallest real number such that there exist k linearly independent vectors of L whose ℓ_i-norms are no greater than $\lambda_k^{(i)}(L)$. We let $\mathrm{Span}(L)$ denote the real vector space spanned by the vectors of L. For a point $t \in \mathrm{Span}(L)$, we let $\mathrm{dist}^{(i)}(t, L) = \inf_{v \in L} \|t - v\|_i$ be the minimal distance between t and any point of L. We define the covering radius of L as $\mu^{(i)}(L) = \sup_{t \in \mathrm{Span}(L)} \mathrm{dist}^{(i)}(t, L)$. When there is no ambiguity, we write $\lambda_k(L)$ and $\mu(L)$ for the successive minima and the covering radius in ℓ_2-norm. The determinant (or volume) $\det(L)$ of a full-rank lattice L is the absolute value of the determinant of any of its bases.

Lemma 2.2 (Minkowski's inequality)**.** *For any full-rank lattice L of dimension n, we have $\lambda_1^{(\infty)}(L) \leq \det(L)^{1/n}$. This implies that $\lambda_1^{(2)}(L) \leq \sqrt{n} \cdot \det(L)^{1/n}$.*

2.2.1 Algorithmic problems

We will consider the following algorithmic problems involving lattices.

Definition 2.3 (Approximate Shortest Vector Problem (approx-SVP))**.** Given a lattice L and $i \in \{1, 2, \infty\}$, the approximate Shortest Vector Problem in norm ℓ_i, with approximation factor $\gamma \geq 1$, is to find a vector $v \in L \setminus \{0\}$ such that $\|v\|_i \leq \gamma \cdot \lambda_1^{(i)}(L)$.

Definition 2.4 (Approximate Closest Vector Problem (approx-CVP))**.** Given a lattice L, $i \in \{1, 2, \infty\}$ and a target $t \in \mathrm{Span}(L)$, the approximate Closest Vector Problem in norm ℓ_i, with approximation factor $\gamma \geq 1$, is to find a vector $v \in L$ such that $\|v - t\|_i \leq \gamma \cdot \mathrm{dist}^{(i)}(t, L)$.
In this thesis, we will be essentially interested in a variant of approx-CVP, in which we ask that $\|v - t\|_i \leq \beta$ for some β, independently of $\mathrm{dist}^{(i)}(t, L)$ (i.e., the distance of the found vector is bounded in absolute terms, independently of whether the target is close to the lattice or not). We call this variant approx-CVP'. For $i \in \{1, 2, \infty\}$, we let $T_{\mathrm{CVP}}(i, L, \beta)$ denote the worst-case run-time of the best known algorithm that solves approx-CVP' for the ℓ_i-norm, in the lattice L, with a bound β.

Definition 2.5 (Approx-CVP with Pre-processing (approx-CVPP))**.** This problem is the same as approx-CVP, except that the algorithm can perform some pre-processing on the lattice L before it gets the target vector t. Approx-CVPP' is defined analogously. We will then consider the pre-processing time (performed when knowing only L) and the query time (performed once we get the target t). For $i \in \{1, 2, \infty\}$, we let $T_{\mathrm{CVP}}^{\text{pre-proc}}(i, L, \beta)$ (resp. $T_{\mathrm{CVP}}^{\text{query}}(i, L, \beta)$) denote the worst-case run-time of the pre-processing phase (resp. query phase) of the best algorithm that solves approx-CVPP' for the ℓ_i-norm, in the lattice L, with a bound β.

In the following, we will always be interested in the approximate versions of these problems, so we will sometimes omit the 'approx' prefix.

In [Laa16], Laarhoven gives a heuristic algorithm for solving approx-CVPP. The following result is not explicitly stated in [Laa16] (only the two extreme values are given), but the computations can be readily adapted.

Theorem 2.6 (Corollaries 2 and 3 of [Laa16])**.** *Let $\alpha \in [0, 1/2]$. Then, under Heuristic 2.7 below, there exists an algorithm that takes as pre-processing input an n-dimensional lattice L (given by a basis whose bit-size is polynomial in n) and as query input any vector $t \in \mathrm{Span}(L)$ (with bit-size that is polynomial in n) and outputs a vector $v \in L$ such that $\|t - v\|_2 \leq O(n^\alpha) \cdot \mathrm{dist}^{(2)}(t, L)$, with pre-processing time $2^{O(n)}$ and query time $\mathrm{poly}(n) \cdot 2^{O(n^{1-2\alpha})}$ (the memory needed during the query phase is also bounded by $\mathrm{poly}(n) \cdot 2^{O(n^{1-2\alpha})}$).*

The heuristic assumption used in Laarhoven's algorithm states that the lattice L is somehow dense and behaves randomly.

Heuristic 2.7. *There exists a constant $c > 0$ such that the ball of radius $c \cdot \lambda_1^{(2)}(L)$ (in ℓ_2-norm) contains at least 2^n points of L. Moreover, once renormalized, these points 'behave' as uniformly and independently distributed points on the unit sphere.*

We can weaken this heuristic assumption by taking $c = \text{poly}(\log n)$, in which case the approximation factor in Laarhoven's algorithm becomes $\widetilde{O}(n^\alpha)$ (the pre-processing and query costs remain the same). We will use this algorithm to heuristically solve approx-CVPP' in Euclidean norm for $\alpha \in [0, 1/2]$, achieving $T_{\text{CVP}}^{\text{pre-proc}}(2, L, O(n^\alpha) \cdot \mu^{(2)}(L)) = 2^{O(n)}$ and $T_{\text{CVP}}^{\text{query}}(2, L, O(n^\alpha) \cdot \mu^{(2)}(L)) = 2^{\widetilde{O}(n^{1-2\alpha})}$.

Babai's roundoff algorithm [Bab86] can be used to solve approx-CVP' with an approximation factor depending on the quality of the basis we have. Assume that we are given a basis $(b_1, \ldots, b_n)$ of a full-rank lattice $L \subset \mathbb{R}^n$. Let $t \in \mathbb{R}^n$ be a target vector. Babai's roundoff algorithm proceeds as follows. Define $B \in \mathbb{R}^{n \times n}$ the matrix whose columns are the b_i's and output $s = B\lfloor B^{-1}t \rceil$, where the rounding is performed coefficient-wise. One can see that such an algorithm outputs a solution $s \in L$ satisfying $\|s - t\| \leq \sum_i \|b_i\|$. This algorithm is typically used, when we know a short basis of a lattice, to shorten vectors modulo the lattice.

2.3 Number fields

A number field K is defined as $K := \mathbb{Q}[X]/(P(X))$ for some irreducible polynomial $P \in \mathbb{Q}[X]$. The degree n of P is called the degree of the number field K. The ring of integers R of the number field K is the set of all the elements of K which can be annihilated by a monic polynomial with integer coefficients. In the rest of this thesis, all number fields K will have degree n.

If P is the m-th cyclotomic polynomial (i.e., its roots are primitive m-th roots of unity), then $\mathbb{Q}[X]/(P(X))$ is called a cyclotomic number field of conductor m. If m is a power of a prime number then $\mathbb{Q}[X]/(P(X))$ is called a prime power cyclotomic field. The number field K used in Chapters 3 and 4 is an arbitrary number field of degree n. In Chapters 5 and 6, the number field used is always a cyclotomic field of conductor $2n$, with n a power of two. In this case, the degree of K is again n. Moreover, K and R can be defined as $K = \mathbb{Q}[X]/(X^n + 1)$ and $R = \mathbb{Z}[X]/(X^n + 1)$.

2.3.1 Embeddings

A number field K of degree n comes with n embeddings from K to $\mathbb{C}$ (i.e., n field morphisms from K to $\mathbb{C}$). We call them $\sigma_1, \ldots, \sigma_n$. Let r_1 be the number of real embeddings (i.e., embeddings arriving in $\mathbb{R} \subset \mathbb{C}$). The remaining $n - r_1$ embeddings are called complex embeddings. There are an even number of them, as they come in pairs of conjugates. Let r_2 be the number of pairs of complex embeddings, so that $n = r_1 + 2r_2$. We order the embeddings such that $\sigma_1, \ldots, \sigma_{r_1}$ are the real embeddings and $\sigma_{r_1+i} = \overline{\sigma_{r_1+r_2+i}}$ for all $1 \leq i \leq r_2$.

Each embedding σ_i from K to $\mathbb{C}$ corresponds to evaluation in one of the roots α_i of P. More formally, let $x \in K$ and $Q \in \mathbb{Q}[X]$ be any representative of x in $\mathbb{Q}[X]$. Then $\sigma_i(x) = Q(\alpha_i)$ (observe that this does not depend on the choice of the representative Q because α_i is a root of P). The real embeddings correspond to the real roots of P whereas the pairs of complex embeddings correspond to the pairs of complex roots of P.

The algebraic norm of an element $x \in K$ is defined by $\mathcal{N}(x) = \prod_{i \leq n} \sigma_i(x)$. Observe that the algebraic norm is multiplicative, i.e., $\mathcal{N}(xy) = \mathcal{N}(x)\mathcal{N}(y)$ for any $x, y \in K$.

2.3.2 Geometry

A number field K of degree n can be embedded into $\mathbb{R}^n$ in two main ways. The canonical embedding (or Minkowski embedding) is defined as

$$\sigma(x) = (\sigma_1(x), \ldots, \sigma_{r_1+r_2}(x)) \in \mathbb{R}^{r_1} \times \mathbb{C}^{r_2},$$

for any $x \in K$. The codomain $\mathbb{R}^{r_1} \times \mathbb{C}^{r_2}$ is itself often embedded into $\mathbb{R}^n$ by taking the real and imaginary parts of the complex coefficients (note that this transformation preserves the ℓ_2-norm). The coefficient embedding is defined by

$$\sigma_{\text{coeff}}(x) = (q_0, \ldots, q_{n-1}),$$

where $Q = \sum_i q_i X^i$ is the unique representative of x of degree less than n. In Chapters 3 and 4, the elements of K are always considered with respect to their canonical embedding and in Chapters 5 and 6 they are always considered with respect to their coefficient embedding. We will often abuse notation and let $x \in K$ refer to both the element of K and its embedding $\mathbb{R}^n$ (which, again, is the canonical embedding for Chapters 3 and 4 and the coefficient embedding for Chapters 5 and 6).

The embedding of K into $\mathbb{R}^n$ defines a geometry over K, induced by the geometry of $\mathbb{R}^n$. The embedding of R into $\mathbb{R}^n$ produces a lattice of rank n. The geometry of this lattice differs between the canonical and the coefficients embedding. In the case of the canonical embedding, we have the following bounds on the norm of a product.

$$\|\sigma(xy)\|_\infty \leq \|\sigma(x)\|_\infty \cdot \|\sigma(y)\|_\infty \tag{2.4}$$

$$\|\sigma(xy)\| \leq \|\sigma(x)\|_\infty \cdot \|\sigma(y)\| \leq \|\sigma(x)\| \cdot \|\sigma(y)\|. \tag{2.5}$$

We do not have similar upper bounds when using the coefficient embedding because the norm of the product depends on the reduction modulo P, which depends on the defining polynomial P.[1]

2.3.3 The ring $K_\mathbb{R}$

The ring $K_\mathbb{R}$ is defined as the tensoring of K with the real numbers $K_\mathbb{R} := K \otimes_\mathbb{Q} \mathbb{R}$. This ring is isomorphic to $\mathbb{R}[X]/(P(X))$. The canonical embedding also provides a ring isomorphism between $K_\mathbb{R}$ and $\mathbb{R}^{r_1} \times \mathbb{C}^{r_2}$ (where multiplication is performed coefficient-wise). Note that K is a field but $K_\mathbb{R}$ is only a (non-integral) ring. The algebraic norm extends to elements of $K_\mathbb{R}$ with the same definition as in K.

Observe that the group $K_\mathbb{R}^\times$ of invertible elements of $K_\mathbb{R}$ is the subset of vectors whose canonical embedding only has non-zero coordinates. We let $K_\mathbb{R}^+$ denote the subset of vectors in $K_\mathbb{R}$ whose canonical embedding has non-negative real coefficients (i.e., the subset $(\mathbb{R}_{\geq 0})^{r_1+r_2} \subset \mathbb{R}^{r_1} \times \mathbb{C}^{r_2}$). For $x \in K_\mathbb{R}$, we let $\overline{x}$ refer to the element of $K_\mathbb{R}$ obtained by complex conjugation of every coordinate of its canonical embedding. More formally, $\overline{x} = \sigma^{-1}(\overline{\sigma(x)})$, where the conjugation is performed coefficient-wise. Note that even if $x \in K$, then we could have $\overline{x} \in K_\mathbb{R} \setminus K$. We call $x\bar{x} \in K_\mathbb{R}^+$ the autocorrelation of $x \in K_\mathbb{R}$, and write it $A(x)$. We can also define a square-root (resp. k-th root) $\sqrt{\cdot} : K_\mathbb{R}^+ \to K_\mathbb{R}^+$ by taking coordinate-wise square roots (resp. k-th root) of the canonical embedding. For any $x \in K_\mathbb{R}^+$ it holds that $A(\sqrt{x}) = x$. Finally, we define equivalence over $K_\mathbb{R}^+$ up to scaling by reals, and write $x \sim y$ for invertible elements $x, y \in K_\mathbb{R}^+$ if $x = \alpha y$ for some positive real number $\alpha > 0$.

2.3.4 Power-of-two cyclotomic fields

In this section, we give some properties of the power-of-two cyclotomic field $K = \mathbb{Q}[X]/(X^n+1)$ and its ring of integers $R = \mathbb{Z}[X]/(X^n+1)$ (for n a power of two). These properties will be used in Chapters 5 and 6. In this special case, the field K only has complex embeddings, hence we have $r_1 = 0$ and $r_2 = n/2$. Further, the geometry induced by the coefficient embedding is the same, up to rotation and scaling, as the one induced by the canonical embedding. More precisely, for any $x, y \in K$, we have

$$\langle \sigma(x), \sigma(y) \rangle = n/2 \cdot \langle \sigma_{\text{coeff}}(x), \sigma_{\text{coeff}}(y) \rangle.$$

In particular, for all $x \in K$, we have

$$\|\sigma(x)\| = \sqrt{n/2} \cdot \|\sigma_{\text{coeff}}(x)\|. \tag{2.6}$$

This equation implies in particular that for any $x \in K$, we have $\|\sigma_{\text{coeff}}(x)\| \leq \sqrt{2}\|\sigma(x)\|_\infty$. Recall that in Chapters 5 and 6, we are going to consider a power-of-two cyclotomic ring with respect to the

[1] We note that the canonical embedding σ also depends on the polynomial P, but once we are given $\sigma(x)$ and $\sigma(y)$, the product $\sigma(xy)$ can be computed without knowing P, which is not the case for the coefficient embedding.

coefficient embedding. With what we have just said, this is equivalent (up to scaling) to the canonical embedding. Hence, we could also have used the canonical embedding. However, the literature on multilinear maps and obfuscation almost exclusively uses the coefficient embedding, hence we kept it for consistency.

In the case of a power-of-two cyclotomic ring, we can bound the norm of a product of elements in coefficient embeddings. For any $x, y \in K$, we have

$$\|\sigma_{\text{coeff}}(xy)\| \leq \sqrt{n} \cdot \|\sigma_{\text{coeff}}(x)\| \cdot \|\sigma_{\text{coeff}}(y)\|, \tag{2.7}$$

$$\|\sigma_{\text{coeff}}(xy)\|_\infty \leq \|\sigma_{\text{coeff}}(x)\|_\infty \cdot \|\sigma_{\text{coeff}}(y)\|_1. \tag{2.8}$$

The complex conjugation is an automorphism of R and K, sending X to X^{-1} (recall that in the general case, complex conjugation is defined over $K_{\mathbb{R}}$ and can send an element of K outside K). More precisely, if $x = x_0 + x_1 X + x_2 X^2 + \cdots + x_{n-1} X^{n-1} \in K_{\mathbb{R}}$, then $\overline{x} = x_0 - x_{n-1} X - x_{n-2} X^2 - \cdots - x_1 X^{n-1}$. This gives us the following equations, in coefficient embedding (the first one is obtained by looking at the constant coefficient of $x\bar{x}$).

$$\|\sigma_{\text{coeff}}(x)\|^2 \leq \|\sigma_{\text{coeff}}(x\bar{x})\|_\infty \tag{2.9}$$

$$\|\sigma_{\text{coeff}}(\bar{x})\| = \|\sigma_{\text{coeff}}(x)\| \text{ and } \|\sigma_{\text{coeff}}(\bar{x})\|_\infty = \|\sigma_{\text{coeff}}(x)\|_\infty. \tag{2.10}$$

Let q be a positive integer. We let R_q denote the quotient ring $R_q := R/(qR) = (\mathbb{Z}/q\mathbb{Z})[X]/(X^n+1)$. For $x \in R$, we write $[x]_q$ (or $[x]$ when there is no ambiguity) the coset of the element x in R_q. We will often lift back elements from R_q to R, in which case we may implicitly mean that we choose the representative with coefficients in the range $[-q/2, q/2]$. If $q \geq 3$ is prime, then $X^n + 1$ factors modulo q as a product of k coprime polynomials of the same degree n/k for some $k|n$. In this case, we have $R_q = \mathbb{F}_q[X]/(P) \simeq (\mathbb{F}_{q^{n/k}})^k$ by the Chinese remainder theorem. If in addition $q \equiv 1 \bmod 2n$, then we know that $k = n$ and so $R_q \simeq (\mathbb{F}_q)^n$.

2.3.5 Discriminant

The discriminant Δ of a number field K is defined by $\Delta = [\det(\sigma_i(r_j))_{i,j}]^2$ for $r_1, \ldots, r_n$ any basis of the $\mathbb{Z}$-module R. When R is seen as a lattice of $\mathbb{R}^n$ via the canonical embedding, then the volume of R as a lattice is related to the discriminant of K by the formula $\det(R) = 2^{-r_2}\sqrt{|\Delta|}$. Minkowski's bound gives us the following inequality between the discriminant and the degree of the number field:

$$\log |\Delta| \geq \Omega(n). \tag{2.11}$$

We will repeatedly use this inequality to simplify cost estimates.

2.3.6 Ideals

A fractional ideal I of K is a subset of K which is stable by addition, and by multiplication with any element of R, and such that $dI \subseteq R$ for some $d \in \mathbb{Z} \setminus \{0\}$. An ideal I is said to be integral if it is contained in R. A non-zero fractional ideal $I \subseteq R$ can be seen as a full-rank lattice in $\mathbb{R}^n$, via the canonical or the coefficients embedding (the geometry of the ideal will differ depending on the choice of the embedding). For an element $g \in K$, we write $\langle g \rangle = gR$ the smallest fractional ideal containing g. Such an ideal is said to be principal. An integral ideal $I \subseteq R$ is said to be prime if the ring R/I is an integral domain. The product of two fractional ideals I and J is defined by $I \cdot J = \{x_1 y_1 + \cdots + x_r y_r \mid r \geq 0, x_1, \ldots, x_r \in I, y_1, \ldots, y_r \in J\}$.

The algebraic norm $\mathcal{N}(I)$ of a non-zero fractional ideal $I \subseteq R$ is the determinant of I when seen as a lattice in $\mathbb{R}^n$ (via the canonical embedding), divided by $\det(R) = 2^{-r_2}\sqrt{|\Delta|}$ (and $\mathcal{N}(\langle 0 \rangle)$ is defined as 0). If I is integral, this is also equal to $|R/I|$. The algebraic norm of a prime ideal is a power of a prime number. For two fractional ideals I and J, the algebraic norm of their product satisfies $\mathcal{N}(I \cdot J) = \mathcal{N}(I) \cdot \mathcal{N}(J)$. For any element $r \in R$, we have that $\mathcal{N}(\langle r \rangle) = |\mathcal{N}(r)|$.

Let I be a non-zero fractional ideal seen as a lattice via the canonical embedding. By definition of the norm of I and Minkowski's inequality, we know that $\lambda_1^{(\infty)}(I) \leq \mathcal{N}(I)^{1/n} \cdot |\Delta|^{1/(2n)}$. We also have the following lower bound

$$\lambda_1^{(\infty)}(I) \geq \mathcal{N}(I)^{1/n}. \tag{2.12}$$

This lower bound comes from the fact that if $x \in I$, then we have $|\mathcal{N}(x)| = \prod_i |\sigma_i(x)| \geq \mathcal{N}(I)$ (because $\langle x \rangle$ is a sub-lattice of I). This implies that at least one of the $|\sigma_i(x)|$'s is no smaller than $\mathcal{N}(I)^{1/n}$. The inequality is obtained by taking x such that $\|\sigma(x)\|_\infty = \lambda_1^{(\infty)}(I)$. When $\log |\Delta| = \widetilde{O}(n)$, these two inequalities imply that $\lambda_1^{(\infty)}(I)$ is essentially $\mathcal{N}(I)^{1/n}$, up to a $2^{\mathrm{poly}(\log n)}$ factor. When $|\Delta|$ increases, so does the gap between the two bounds.

2.3.7 Modules

In this thesis, we call (R-)module any set of the form $M = I_1\mathbf{b}_1 + \ldots + I_m\mathbf{b}_m$, where the I_j's are non-zero fractional ideals of R and the $\mathbf{b}_j$'s are $K_\mathbb{R}$-linearly independent vectors in $K_\mathbb{R}^s$, for some $s > 0$.[2] The tuple of pairs $((I_1, \mathbf{b}_1), \ldots, (I_m, \mathbf{b}_m))$ is called a pseudo-basis of M, and m is its rank. Note that the notion of rank of a module is usually only defined when the module has a basis (i.e., is of the form $M = R\mathbf{b}_1 + \ldots + R\mathbf{b}_m$, with all the ideals equal to R). In this thesis, we consider an extension of the definition of rank, defined even if the module does not have a basis, as long as it has a pseudo-basis. In particular, fractional ideals are rank-1 modules contained in K, and sets of the form $\alpha \cdot I$ for $\alpha \in K_\mathbb{R}^\times$ and a non-zero fractional ideal I are rank-1 modules in $K_\mathbb{R}$. We refer to [Hop98] for a thorough study of R-modules, and concentrate here on the background necessary to Chapter 4. When dealing with modules, we use bold letters to refer to vectors with coefficients in $K_\mathbb{R}$.

Two pseudo-bases $((I_1, \mathbf{b}_1), \ldots, (I_m, \mathbf{b}_m))$ and $((J_1, \mathbf{c}_1), \ldots, (J_m, \mathbf{c}_m))$ represent the same module if and only if there exists $\mathbf{U} = (u_{ij})_{i,j} \in K^{m \times m}$ invertible such that $\mathbf{C} = \mathbf{B} \cdot \mathbf{U}$; we have $u_{ij} \in I_i J_j^{-1}$ and $u'_{ij} \in J_i I_j^{-1}$ for all i, j and for $\mathbf{U}' = (u'_{ij})_{i,j} := \mathbf{U}^{-1}$. Here, the matrix $\mathbf{B}$ is the concatenation of the column vectors $\mathbf{b}_i$ (and similarly for $\mathbf{C}$). If $m > 0$, we define $\det_{K_\mathbb{R}} M = \det(\overline{\mathbf{B}}^\top \mathbf{B})^{1/2} \cdot \prod_i I_i$. It is an R-module in $K_\mathbb{R}$. Note that it is a module invariant, i.e., it is identical for all pseudo-bases of M.

We extend the canonical embedding to vectors $\mathbf{v} = (v_1, \ldots, v_s)^T \in K_\mathbb{R}^s$ by defining $\sigma(\mathbf{v})$ as the vector of $\mathbb{R}^{ns}$ obtained by concatenating the canonical embeddings of the v_i's. This extension of the canonical embedding maps any module M of rank m to a (nm)-dimensional lattice in $\mathbb{R}^{ns}$. We abuse notation and use M to refer to both the module and the lattice obtained by applying the canonical embedding. The determinant of a module M seen as a lattice is $\det M = 2^{-r_2 m} \Delta_K^{m/2} \cdot \mathcal{N}(\det_{K_\mathbb{R}} M)$.

We consider the following inner product for $\mathbf{a}, \mathbf{b} \in K_\mathbb{R}^s$:

$$\langle \mathbf{a}, \mathbf{b} \rangle_{K_\mathbb{R}} = \sum_{i \in [s]} \overline{a}_i b_i \in K_\mathbb{R}$$

Note that we have $\langle \mathbf{v}, \mathbf{v} \rangle_{K_\mathbb{R}} \in K_\mathbb{R}^+$, as all $\sigma_i(\langle \mathbf{v}, \mathbf{v} \rangle_{K_\mathbb{R}})$'s are non-negative. For $\mathbf{v} \in K_\mathbb{R}^s$, we define $\|\mathbf{v}\|_{K_\mathbb{R}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{K_\mathbb{R}}} \in K_\mathbb{R}$ and $\|\mathbf{v}\| = \sqrt{\sum_{1 \leq j \leq r_1 + r_2} \sigma_j(\langle \mathbf{v}, \mathbf{v} \rangle_{K_\mathbb{R}})} \in \mathbb{R}_{\geq 0}$. Observe that if we write $\mathbf{v} = (v_1, \ldots, v_s)$, then $\|\mathbf{v}\| = \sqrt{\sum_{1 \leq j \leq r_1 + r_2} \sum_{1 \leq i \leq s} |\sigma_j(v_i)|^2} = \|\sigma(\mathbf{v})\|$. Hence, this notation matches the usual Euclidean norm when $\mathbf{v}$ is seen as a vector in R^{ns} via the canonical embedding. We extend the infinity norm to vectors $\mathbf{v} \in K_\mathbb{R}^s$ by $\|\mathbf{v}\|_\infty = \max_{i \in [s]} \|v_i\|_\infty$, where $\mathbf{v} = (v_1, \ldots, v_m)$. We also extend the algebraic norm to vectors $\mathbf{v} \in K_\mathbb{R}^s$ by setting $\mathcal{N}(\mathbf{v}) := \mathcal{N}(\|\mathbf{v}\|_{K_\mathbb{R}})$. For $s = 1$, we see that $\mathcal{N}(\mathbf{v}) = |\mathcal{N}(\mathbf{v})|$. By applying the arithmetic-geometric inequality to $\sigma_1(\langle \mathbf{a}, \mathbf{a} \rangle), \cdots, \sigma_n(\langle \mathbf{a}, \mathbf{a} \rangle)$ and observing that $\sum_{i=1}^n \sigma_i(\langle \mathbf{a}, \mathbf{a} \rangle) \leq 2 \sum_{i=1}^{r_1 + r_2} \sigma_i(\langle \mathbf{a}, \mathbf{a} \rangle)$, we obtain that $\mathcal{N}(\mathbf{a})^{1/n} \leq \sqrt{2/n} \cdot \|\mathbf{a}\|$ for $\mathbf{a} \in K_\mathbb{R}^s$.

We define the module minimum $\lambda_1(M)$ as the norm of a shortest non-zero element of M with respect to $\|\cdot\|$. The module-LLL algorithm described in Chapter 4 will rely on the algebraic norm rather than the Euclidean norm. For this reason, we will also be interested in the minimum $\lambda_1^\mathcal{N}(M) = \min(\mathcal{N}(\mathbf{v}) : \mathbf{v} \in M \setminus \{\mathbf{0}\})$. The following lemma provides relationships between $\lambda_1(M)$ and $\lambda_1^\mathcal{N}(M)$.

Lemma 2.8. *For any rank-m module M, we have:*

$$\lambda_1(M)^n \Delta_K^{-1/2} n^{-n/2} \leq \lambda_1^\mathcal{N}(M) \leq 2^{n/2} n^{-n/2} \lambda_1(M)^n \leq 2^{n/2} m^{n/2} \Delta_K^{1/2} \mathcal{N}(\det_{K_\mathbb{R}} M)^{1/m}.$$

Proof. Let $\mathbf{s} \in M \setminus \{\mathbf{0}\}$ of minimal Euclidean norm. By the arithmetic-geometric inequality, we have $\mathcal{N}(\mathbf{s}) \leq 2^{n/2} n^{-n/2} \|\mathbf{s}\|^n$. By Minkowski's theorem applied to the canonical embedding of M,

[2] The vectors $\mathbf{b}_j$'s are said to be $K_\mathbb{R}$-linearly independent if and only if there is no non-trivial way to write the zero vector as a $K_\mathbb{R}$-linear combination of the $\mathbf{b}_j$'s. Because $K_\mathbb{R}$ is a ring and not a field, this definition is stronger than requiring that none of the $\mathbf{b}_j$'s is in the span of the others.

we have $\|\mathbf{s}\| \leq \sqrt{mn}(\det M)^{1/(mn)}$. Using the inequality $\det M \leq \Delta_K^{m/2}\mathcal{N}(\det_{K_\mathbb{R}} M)$ allows to obtain the last two inequalities. The first inequality follows from taking $\mathbf{s} \in M \setminus \{\mathbf{0}\}$ of minimal algebraic norm and applying Minkowski's theorem to the lattice $\sigma(R \cdot \mathbf{s})$. □

2.3.7.1 Gram-Schmidt orthogonalization of module lattices

We extend Gram-Schmidt Orthogonalization from matrices over the real numbers to matrices over $K_\mathbb{R}$. For $(\mathbf{b}_1, \ldots, \mathbf{b}_m) \in K_\mathbb{R}^{s\times m}$ such that $\sum_i R\mathbf{b}_i$ is a module of rank m, we define $\mathbf{b}_1^* = \mathbf{b}_1$ and, for $1 < i \leq m$:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j<i} \mu_{ij}\mathbf{b}_j^* \text{ with } \forall j < i : \mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle_{K_\mathbb{R}}}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle_{K_\mathbb{R}}}.$$

It may be checked that $\langle \mathbf{b}_i^*, \mathbf{b}_j^* \rangle_{K_\mathbb{R}} = 0$ for $i \neq j$, and that $\mathbf{b}_i^* = \text{argmin}(\|\mathbf{b}_i - \sum_{j<i} y_j\mathbf{b}_j\| | \forall j : y_j \in K_\mathbb{R})$.

We also extend the QR-factorization to matrices over $K_\mathbb{R}$. We define $r_{ii} = \|\mathbf{b}_i^*\|_{K_\mathbb{R}}$ for $i \leq m$, $r_{ij} = \mu_{ji}r_{ii}$ when $i < j$, and $r_{ij} = 0$ when $i > j$. We then have $\mathbf{B} = \mathbf{Q} \cdot \mathbf{R}$, where $\mathbf{Q} \in K_\mathbb{R}^{s\times m}$ is the matrix whose columns are the $\mathbf{b}_i^*/\|\mathbf{b}_i^*\|_{K_\mathbb{R}}$'s and $\mathbf{R} = (r_{ij})_{ij}$. Note that $\overline{\mathbf{Q}}^T\mathbf{Q} = \mathbf{Id}$ and that $\mathbf{R}$ is upper-triangular with diagonal coefficients in $K_\mathbb{R}^+$.

The following lemma provides relationships between some module invariants and the QR-factorization.

Lemma 2.9. *Let M be a module with pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq m}$. Let $\mathbf{R}$ denote the R-factor of $\mathbf{B}$. We have $\det_{K_\mathbb{R}} M = \prod_i r_{ii}I_i$ and $\det M = 2^{-r_2 m}\Delta_K^{m/2}\prod_i \mathcal{N}(r_{ii}I_i)$. Further, we have that $\lambda_1^{\mathcal{N}}(M) = \min_{\mathbf{s}\in M\setminus\{\mathbf{0}\}} \mathcal{N}(\mathbf{s}) \geq \min_i \mathcal{N}(r_{ii}I_i)$.*

Proof. Recall that we have $\det_{K_\mathbb{R}} M = (\det \overline{\mathbf{B}}^\top\mathbf{B})^{1/2} \cdot \prod_i I_i$. Using the QR-decomposition (and in particular the facts that $\overline{\mathbf{Q}}^T\mathbf{Q} = \mathbf{Id}$ and that $\mathbf{R}$ is upper-triangular with diagonal coefficients in $K_\mathbb{R}^+$), this rewrites as $\det_{K_\mathbb{R}} M = (\det \overline{\mathbf{R}}^\top\mathbf{R})^{1/2} \cdot \prod_i I_i = \prod_i r_{ii}I_i$. The equality $\det M = 2^{-r_2 m}\Delta_K^{m/2}\mathcal{N}(\det_{K_\mathbb{R}} M)$ leads to the first statement.

For the second statement, note that for any $\mathbf{v} \in K_\mathbb{R}^m$, we have $\|\mathbf{Bv}\|_{K_\mathbb{R}} = \|\overline{\mathbf{Q}}^T\mathbf{Bv}\|_{K_\mathbb{R}} = \|\mathbf{Rv}\|_{K_\mathbb{R}}$. This implies that $\mathcal{N}(\mathbf{Bv}) = \mathcal{N}(\mathbf{Rv})$. Therefore, it suffices to prove the result for the module spanned by the pseudo-basis $((I_i, \mathbf{r}_i))_{i\leq m}$, where the $\mathbf{r}_i$'s are the columns of $\mathbf{R}$. Take $\mathbf{s} = \sum x_i\mathbf{r}_i \in M \setminus \{\mathbf{0}\}$ (with $x_i \in I_i$ for every $i \leq m$), and consider $i_0 = \max(i | \forall j > i : x_j = 0)$. Then, by using the triangular shape of $\mathbf{R}$, we obtain that $\mathcal{N}(\mathbf{s}) \geq \mathcal{N}(x_{i_0}r_{i_0 i_0})$. The proof can be completed by noting that $\mathcal{N}(x_{i_0}) \geq \mathcal{N}(I_{i_0})$. □

For lattices, if we have a basis and a full-rank family of short vectors, then we can efficiently obtain a basis of the lattice whose Gram-Schmidt vectors are no longer than those of the full-rank family of short vectors. This was generalized to modules in [FS10], relying on the extension to modules of the Hermite Normal Form [BP91, Coh96, BFH17].

Lemma 2.10 (Theorem 4 of [FS10])**.** *There exists an algorithm that takes as inputs a pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq m}$ of a module $M \subset K_\mathbb{R}^s$ and a full-rank set of vectors $(\mathbf{s}_i)_{i\leq m}$ of M and outputs a pseudo-basis $((J_i, \mathbf{c}_i))_{i\leq m}$ such that $\mathbf{c}_i \in M$ and $\mathbf{c}_i^* = \mathbf{s}_i^*$ for all i. If $M \subset K^s$, then it terminates in polynomial-time.*

Note that the condition that $\mathbf{c}_i \in M$ implies that $R \subseteq J_i$, for all i.

2.3.8 The class group

We let $\mathcal{I}_K$ denote the set of non-zero fractional ideals of K and $\mathcal{P}_K \subseteq \mathcal{I}_K$ denote the subset of non-zero principal fractional ideals. One can prove that for every non-zero fractional ideal I, there is a fractional ideal I^{-1} such that $I \cdot I^{-1} = R$. This gives $\mathcal{I}_K$ a group structure, for which $\mathcal{P}_K$ is a subgroup.

The class group of K is defined as the quotient $Cl_K = \mathcal{I}_K/\mathcal{P}_K$. For any non-zero ideal I of K, we let $[I]$ denote the equivalence class of I in the class group. In particular, we have $\mathcal{P}_K = [R]$. The class group is a finite abelian group and its cardinality h_K is called the class number. We have the following bound:

$$\log h_K = \widetilde{O}(\log|\Delta|). \tag{2.13}$$

This can be derived from the proof of Equation (2.11), this proof being based on the fact that any class of the class group contains an integral ideal whose norm is bounded as $2^{\widetilde{O}(\log|\Delta|)}$. We also justify it later using Equation (2.14) (which is significantly stronger).

We know, thanks to a result of Bach [Bac90] that the class group can be generated by ideals of polynomially bounded norms.

Theorem 2.11 (Theorem 4 of [Bac90])**.** *Under the GRH, the class group of a number field of discriminant Δ is generated by the prime ideals of algebraic norms $\leq 12\log^2|\Delta|$.*

Moreover, computing all prime ideals of norms $\leq 12\log^2|\Delta|$ can be done in time polynomial in $\log|\Delta|$. Indeed, these prime ideals can be obtained by factoring all ideals $\langle p\rangle$ where $p \in \mathbb{Z}$ is a prime no greater than $12\log^2\Delta$. Further, factoring such an ideal can be done in polynomial time (see [Coh13, Section 4.8.2 and 6.2]).

Lemma 2.12. *Let $\mathfrak{B}$ be any finite set of fractional ideals that generates the class group Cl_K. Then we can extract a subset $\mathfrak{B}'$ of $\mathfrak{B}$, of cardinality at most $\log h_K$, which also generates the class group. Moreover, this can be done efficiently if we are given the relations between the elements of $\mathfrak{B}$, in the form of a basis of $\ker(f_{\mathfrak{B}})$ where $f_{\mathfrak{B}} : (e_1, \dots e_r) \in \mathbb{Z}^r \mapsto \prod_i [\mathfrak{p}_i^{e_i}] \in Cl_K$, with $\mathfrak{B} = \{\mathfrak{p}_1, \dots, \mathfrak{p}_r\}$.*

Proof. We know that $\ker(f_{\mathfrak{B}})$ is a lattice of volume h_K contained in $\mathbb{Z}^r$ (it is stable by addition and subtraction, and $|\mathbb{Z}^r/\ker(f_{\mathfrak{B}})| = |Cl_K| = h_K$). Let $R_{\mathfrak{B}} \in \mathbb{Z}^{r\times r}$ be a basis of this lattice, with column vectors. From this basis, we can efficiently compute the Hermite Normal Form (HNF) of the lattice, which we will write $H_{\mathfrak{B}}$. This basis matrix is triangular, and each column corresponds to a relation between the elements of $\mathfrak{B}$ (each row corresponds to an ideal of $\mathfrak{B}$). So we can remove from the set $\mathfrak{B}$ any ideal whose row in $H_{\mathfrak{B}}$ has a 1 on the diagonal. Indeed, if row i has a 1 on the diagonal, this means that we have a relation of the form $[\mathfrak{p}_i \cdot \prod_{j>i} \mathfrak{p}_j^{e_j}] = [R]$. Hence the ideal class $[\mathfrak{p}_i]$ is in the group generated by $\{[\mathfrak{p}_j]\}_{j>i}$, and so it is not needed to generate the class group. But we know that $\det(H_{\mathfrak{B}}) = \det(\ker(f_{\mathfrak{B}})) = h_K$ is the product of the diagonal elements (which are integers). So we have at most $\log h_K$ ideals with diagonal entries different from 1. Hence, after removing from $\mathfrak{B}$ all ideals whose corresponding row in $H_{\mathfrak{B}}$ has a 1 on the diagonal, we obtain a set $\mathfrak{B}'$ of cardinality at most $\log h_K$ and which still generates the class group. This proof is an efficient algorithm if we are given an initial basis $R_{\mathfrak{B}}$, because we only need to compute an HNF basis, which can be done in time polynomial in the size of the input matrix. □

Theorem 2.11 states that the class group can be generated by integral ideals of polynomially bounded norms, but this does not give us the existence of many small-norm integral ideals. For instance, if the class group is trivial (i.e., all ideals are principal), then it is generated by $[R]$. More generally, the class group could be generated by a very small number of ideals. In the following chapters, we will need the existence of $\widetilde{\Omega}(\log|\Delta|)$ distinct integral ideals of polynomially bounded norms.

Theorem 2.13 (Theorem 8.7.4 of [BS96])**.** *Assume the GRH. Let $\pi_K(x)$ be the number of prime integral ideals of K of norm $\leq x$. Then there exists an absolute constant C (independent of K and x) such that*

$$|\pi_K(x) - \mathrm{li}(x)| \leq C \cdot \sqrt{x}\,(n\log x + \log|\Delta|)\,,$$

where $\mathrm{li}(x) = \int_2^x \frac{\mathrm{d}t}{\ln t} \sim \frac{x}{\ln x}$ (and $\ln$ refers to the natural logarithm).

Instantiating this theorem with $x = (\log|\Delta|)^\kappa$ for some constant $\kappa > 2+2\varepsilon$, we obtain the following corollary.

Corollary 2.14. *Assume the GRH. Let $\varepsilon > 0$ and $\kappa > 2+2\varepsilon$. For $\log|\Delta|$ sufficiently large, there are at least $(\log|\Delta|)^{\kappa-2\varepsilon}$ distinct prime integral ideals of norm smaller than $(\log|\Delta|)^\kappa$.*

Proof. We apply Theorem 2.13 with $x = (\log|\Delta|)^\kappa$. As $\mathrm{li}(x) \sim \frac{x}{\ln x}$, we have that $\mathrm{li}(x) \geq (\log|\Delta|)^{\kappa-\varepsilon}$ holds for $\log|\Delta|$ sufficiently large. Recall that $\log|\Delta| > cn$ for some (explicit) constant c. Hence, the right hand side of the inequality of Theorem 2.13 can be bounded as

$$C \cdot \sqrt{x}\,(n\log x + \log|\Delta|) \leq C(\kappa/c+1)\cdot(\log|\Delta|)^{\kappa/2+1}\cdot\log\log|\Delta|.$$

But, as we chose κ such that $\kappa - \varepsilon > \kappa/2 + 1$, we have, for $\log|\Delta|$ sufficiently large:

$$(\log|\Delta|)^{\kappa-\varepsilon} - C(\kappa/c+1)\cdot(\log|\Delta|)^{\kappa/2+1}\cdot\log\log|\Delta| \geq (\log|\Delta|)^{\kappa-2\varepsilon},$$

hence proving the corollary. □

We use Theorem 2.11, Corollary 2.14 and Lemma 2.12, to obtain the following.

Corollary 2.15. *Assume the GRH. Then, for* $\log|\Delta|$ *sufficiently large and for any integer* $r \geq \log h_K$*, there exists a set* $\mathfrak{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_r\}$ *of prime integral ideals generating the class group, with* $\mathcal{N}(\mathfrak{p}_i) = \mathrm{poly}(\log|\Delta|, r)$ *for all* i*.*

Proof. Combining Theorem 2.11 and Lemma 2.12, we know that there exists a set $\mathfrak{B}$ of cardinality at most r, generating the class group and containing only prime ideals of norms $\leq 12\log^2|\Delta|$. We can then add prime ideals to this set $\mathfrak{B}$, until its cardinality reaches r. Thanks to Corollary 2.14, we know that there are enough prime ideals of norm smaller than $\mathrm{poly}(\log|\Delta|, r)$ (for some fixed poly) to increase the cardinality of $\mathfrak{B}$ up to r. □

2.3.9 The log-unit lattice

We let $R^\times$ denote the group of units of R, that is $R^\times = \{u \in R \mid \exists v \in R,\ uv = 1\}$. Dirichlet's unit theorem states that $R^\times$ is isomorphic to the Cartesian product of a finite cyclic group (formed by the roots of unity contained in K) with the additive group $\mathbb{Z}^{r_1+r_2-1}$.

We define $\mathrm{Log}\, x = (\log|\sigma_1(x)|, \ldots, \log|\sigma_n(x)|)^T \in \mathbb{R}^n$, for any $x \in K_{\mathbb{R}}^\times$. Observe that this is not the usual definition of the logarithmic embedding. The function Log is often defined either as $(\log|\sigma_1(x)|, \ldots, \log|\sigma_{r_1+r_2}(x)|)^T \in \mathbb{R}^{r_1+r_2}$ [Sam13, Section 4.4] or as $(\log|\sigma_1(x)|, \ldots, \log|\sigma_{r_1}(x)|, 2\log|\sigma_{r_1+1}(x)|, 2\log|\sigma_{r_1+r_2}(x)|)^T \in \mathbb{R}^{r_1+r_2}$ [Coh13, Definition 4.9.6]. Indeed, for $i > r_1 + r_2$, the $\log|\sigma_i(x)|$'s are redundant because $|\sigma_i(x)| = |\sigma_{i-r_2}(x)|$. However, in this thesis, it will be more convenient to work with the logarithms of all the embeddings.

Let $E = \{x \in \mathbb{R}^n : x_i = x_{i+r_2}, \forall r_1 < i \leq r_2\}$. We have $\mathrm{Log}(K_{\mathbb{R}}^\times) \subseteq E$. We let H be the hyperplane of $\mathbb{R}^n$ defined by $H = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 0\}$ and $\mathbf{1}$ be the vector, orthogonal to H, defined as $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^n$. We write $\pi_H : \mathbb{R}^n \to H$ the orthogonal projection on H, parallel to $\mathbf{1}$. We define $\Lambda = \{\mathrm{Log}\, u, u \in R^\times\}$, which is a lattice of dimension $r_1 + r_2 - 1$ contained in $H \cap E$ (thanks to Dirichlet's unit theorem), called the *log-unit lattice*. Its minimum satisfies $\lambda_1(\Lambda) \geq (\ln n)/(6n^2)$ (see [FP06, Cor. 2]). Further, we have the following upper bound:

$$\det(\Lambda)\cdot h_K \leq 2^{O(\log|\Delta| + n\log\log|\Delta|)} = 2^{\widetilde{O}(\log|\Delta|)}. \tag{2.14}$$

This upper bound comes from the relation between $\det(\Lambda)$, h_K and the residue of the zeta-function ζ_K at $s = 1$ (see [Lou00]). The latter is known considering Λ defined by the logarithmic embedding $(\log|\sigma_1(x)|, \ldots, \log|\sigma_{r_1}(x)|, 2\log|\sigma_{r_1+1}(x)|, 2\log|\sigma_{r_1+r_2}(x)|)^T \in \mathbb{R}^{r_1+r_2}$. However, it can be seen that if one multiplies our lattice Λ by a matrix with blocks of the form $\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$ (in order to add and subtract $\log|\sigma_i(x)|$ and $\log|\sigma_{i+r_2}(x)|$ for $r_1 < i \leq r_2$), one obtains the log-unit lattice defined by the logarithmic embedding considered in [Lou00]. As multiplying by such a matrix increases the determinant by a factor 2^{r_2}, Inequality (2.14) remains valid in our setup.

This bound, combined with a lower bound on $\det(\Lambda)$ also gives Equation (2.13). Indeed, using a result of Zimmert [Zim80], we have that $\det(\Lambda) > 0.02 \cdot 2^{-r_2}$ (handling again our unusual definition of Λ).

For any $x \in K_{\mathbb{R}}^\times$, there exists a unique vector $h \in H \cap E$ and a unique real number a such that $\mathrm{Log}\, x = h + a\mathbf{1}$. In the following, we recall relationships between (h, a) and x. These results are standard (e.g., they are used freely in [CDPR16, Section 6]).

Lemma 2.16. *Let* $r \in K$*. Then we have* $\mathrm{Log}\, r = h + \frac{\log|\mathcal{N}(r)|}{n}\mathbf{1}$*, for some* $h \in H \cap E$*.*

For the sake of completeness, and because we are using an unusual definition of Log, we give a proof of this result below.

Proof. Write $\operatorname{Log} r = h + a\mathbf{1}$ for some $h \in H \cap E$ and $a > 0$. First, as $\mathbf{1}$ is orthogonal to H, we have that $\langle \mathbf{1}, \operatorname{Log} r\rangle = \langle \mathbf{1}, a\mathbf{1}\rangle = a \cdot n$. But using the definition of $\operatorname{Log} r$, we also have that

$$\langle \mathbf{1}, \operatorname{Log} r\rangle = \sum_i \log|\sigma_i(r)| = \log|\mathcal{N}(r)|,$$

where we used the fact that $\mathcal{N}(r) = \prod_i \sigma_i(r)$. This completes the proof. □

The following lemma gives a bound on the Euclidean norm of an element $r \in R$ in terms of its decomposition $\operatorname{Log} r = h + a\mathbf{1}$.

Lemma 2.17. *For any $r \in K$, if $\operatorname{Log} r = h + a\mathbf{1}$ with $h \in H \cap E$ and $a \in \mathbb{R}$, then we have $\|r\|_\infty \leq 2^a \cdot 2^{\|h\|_\infty}$. In particular, this implies that*

$$\|r\|_2 \leq \sqrt{n} \cdot 2^a \cdot 2^{\|h\|_2} = \sqrt{n} \cdot |\mathcal{N}(r)|^{1/n} \cdot 2^{\|h\|_2}.$$

Proof. The second inequality follows from the first one by using Equation (2.1) (and Lemma 2.16 for the equality). For the first inequality, recall that by definition of Log, we have that $(\operatorname{Log} r)_i = \log|\sigma_i(r)| = h_i + a$ for all i. So, by definition of $\|r\|_\infty = \max_i |\sigma_i(r)|$, we have $\|r\|_\infty = \max_i 2^{h_i + a} \leq 2^a \cdot 2^{\|h\|_\infty}$. □

2.3.10 Algorithmic problems related to class group computations

Given a basis of a principal ideal $I \subseteq R$ (seen as a sub-lattice of R), the problem of finding a generator of I is known as the principal ideal problem. We let $T_{\text{c-g}}(N)$ denote the best time complexity of an algorithm solving the principal ideal problem for ideals of algebraic norm N.

Let $\mathfrak{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_r\}$ be a set of prime integral ideals generating the class group, obtained for example using Corollary 2.15. We will be interested in computing the lattice of all the relations between the ideals of $\mathfrak{B}$, i.e., the kernel of the map

$$f_{\mathfrak{B}} : e = (e_1, \ldots, e_r) \in \mathbb{Z}^r \mapsto [\prod_i \mathfrak{p}_i^{e_i}] \in Cl_K.$$

Recall that $\ker(f_{\mathfrak{B}})$ is a full-rank sub-lattice of $\mathbb{Z}^r$ of volume $|\mathbb{Z}^r / \ker(f_{\mathfrak{B}})| = |Cl_K| = h_K$. Let $N_{\mathfrak{B}} = \max_i \mathcal{N}(\mathfrak{p}_i)$. We let $T_{\text{rel}}(N_{\mathfrak{B}}, r)$ denote the time needed to compute a basis of $\ker(f_{\mathfrak{B}})$, together with generators of the corresponding principal ideals, given as input the set $\mathfrak{B}$. We write $T_{\text{decomp}}(N, N_{\mathfrak{B}}, r)$ for the time needed, given $\mathfrak{B}$ and a fractional ideal I of norm $\mathcal{N}(I) = N$, to find a vector $e \in \mathbb{Z}^r$ and an element $g \in K$ such that $I = \prod_i \mathfrak{p}_i^{e_i} \cdot \langle g\rangle$. Note that this decomposition always exists but might not be unique (we only require that $\mathfrak{B}$ generates the class group). Finally, we let $T_{\text{log-unit}}$ be the time needed to compute a basis of the log-unit lattice of K.

The four problems above are usually solved by computing S-units[3] for a well-chosen set S. This is why, in the following, the same cost bounds hold for the four of them.

In the *quantum* setting, Biasse and Song [BS16] showed that these four problems can be solved in polynomial time for any number field (under GRH). More precisely, they showed that

- $T_{\text{c-g}}(N) = \text{poly}(n, \log|\Delta|, \log N_{\text{num}}, \log N_{\text{denom}})$;
- $T_{\text{rel}}(N_{\mathfrak{B}}, r) = \text{poly}(\log|\Delta|, r, \log N_{\mathfrak{B}})$;
- $T_{\text{decomp}}(N, N_{\mathfrak{B}}, r) = \text{poly}(\log|\Delta|, \log N_{\text{num}}, \log N_{\text{denom}}, r, \log N_{\mathfrak{B}})$;
- $T_{\text{log-unit}} = \text{poly}(\log|\Delta|)$;

where N_{num} and N_{denom} refer to the numerator and denominator of N (i.e., $N = N_{\text{num}}/N_{\text{denom}} \in \mathbb{Q}$ with $N_{\text{num}}, N_{\text{denom}}$ in $\mathbb{Z}_{>0}$ and coprime).

In the *classical* setting, these four problems can be solved heuristically in sub-exponential time (under GRH). The first sub-exponential algorithm for all number fields (and which allows n to tend to infinity with $\log|\Delta|$) is due to Biasse and Fieker [BF14]:

[3] Given a set $S = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_r\}$ of prime integral ideals, the S-units are the elements $\alpha \in K$ such that there exist $e_1, \ldots, e_r \in \mathbb{Z}$ with $\prod_i \mathfrak{p}_i^{e_i} = \langle \alpha\rangle$.

- $T_{\text{c-g}}(N) = \text{poly}(n, \log N_{\text{num}}, \log N_{\text{denom}}) \cdot 2^{\widetilde{O}((\log|\Delta|)^{2/3})}$;
- $T_{\text{rel}}(N_{\mathfrak{B}}, r) = \text{poly}(r, \log N_{\mathfrak{B}}) \cdot 2^{\widetilde{O}((\log|\Delta|)^{2/3})}$;
- $T_{\text{decomp}}(N, N_{\mathfrak{B}}, r) = \text{poly}(\log N_{\text{num}}, \log N_{\text{denom}}, r, \log N_{\mathfrak{B}}) \cdot 2^{\widetilde{O}((\log|\Delta|)^{2/3})}$;
- $T_{\text{log-unit}} = 2^{\widetilde{O}((\log|\Delta|)^{2/3})}$.

Biasse and Fieker actually claim $2^{O((\log|\Delta|)^{2/3+\varepsilon})}$ run-times. Tracing back the source of this ε leads to Biasse's [Bia17, Proposition 3.1]. A careful reading of the proof of the latter shows that the $(\log|\Delta|)^{\varepsilon}$ term is actually a power of $\log\log|\Delta|$, hence, in our notations, it is absorbed by the $\widetilde{O}$ notation. In addition to the GRH, the algorithm of Biasse and Fieker requires two heuristic assumptions, referred to as Heuristic 1 and Heuristic 3 in [BF14]. We recall these two heuristic assumptions below (see [BF14] for more details).

Heuristic 2.18 (Heuristic 1 of [BF14])**.** *The probability $P(x, y)$ that an integral ideal of R produced by the Biasse-Fieker [BF14] algorithm, of norm bounded by x, can be factored as a product of prime ideals of norms bounded by y satisfies*

$$P(x, y) \geq e^{-(1+o_{x\to\infty}(1))\cdot u\log u} \quad \textit{for} \quad u = \frac{\log x}{\log y}.$$

Heuristic 2.19 (Heuristic 3 of [BF14])**.** *Given a set of r elements generating the class group, the algorithm only needs to find $r^{O(1)}$ relations between these elements to generate the full lattice of relations, with probability close to* 1*.*

Smaller cost bounds are known for specific families of number fields. For prime-power cyclotomic fields, the $2^{\widetilde{O}((\log|\Delta|)^{2/3})}$ bounds can be replaced by $2^{\widetilde{O}((\log|\Delta|)^{1/2})}$ [BEF+17]. This algorithm is again heuristic and relies on the same assumptions as [BF14]. For real multiquadratic number fields, efficient classical algorithms solve these four problems [BBV+17, BV18]. Finally, we note that the exponent 2/3 was recently lowered to 3/5 in [Gel17] and can even be decreased further in some cases.

The short Principal Ideal Problem. Let us conclude by mentioning a last computational problem, which we are going to consider only in power-of-two cyclotomic fields. This problem is a variant of the principal ideal problem, where one of the generators is one of the shortest element of the principal ideal, and we want to recover exactly this generator.

Definition 2.20 (short Principal Ideal Problem (sPIP))**.** Let $h \in R$ be sampled according to some distribution D. The short Principal Ideal Problem is, given any basis of the ideal $\langle h\rangle$ (when seen as a sub-lattice of R), to recover $\pm X^i \cdot h$ for some $i \in \{1, \ldots, n\}$.

For cyclotomic fields of order a power of two, when D is a discrete Gaussian distribution (see below for a definition of Gaussian distributions), this problem can be solved in quantum polynomial time, using the results of [BS16, CGS14a, CDPR16]. Recall that the principal ideal problem can be solved in any number field in polynomial quantum time, hence we can recover a generator $\widetilde{h}$ of $\langle h\rangle$ from its basis. Then, the authors of [CDPR16], based on an observation of [CGS14a], proved that from any generator $\widetilde{h}$ of $\langle h\rangle$, if h has been sampled using a discrete Gaussian distribution, then one can recover $\pm X^i \cdot h$, for some $i \in \{1, \ldots, n\}$, in (classical) polynomial time. This second part (recovering $\pm X^i \cdot h$ from $\widetilde{h}$) relies on the conjecture that the set of cyclotomic units of R is equal to $R^\times$ for power-of-two cyclotomic fields. We summarize this in the following theorem.

Theorem 2.21 (Adapted from [BS16, CDPR16])**.** *Let R be the ring of integers of a power-of-two cyclotomic field. Let $h \in R$ be sampled according to a discrete spherical Gaussian distribution of parameter larger than $200 \cdot n^{1.5}$ (see Section 2.5.2). Then, under Conjecture 2.22 and the Generalized Riemann Hypothesis (GRH), there is a quantum polynomial time algorithm such that, given any basis of the ideal $\langle h\rangle$, it recovers $\pm X^i \cdot h$ for some $i \in \{1, \ldots, n\}$, with constant probability close to 1 over the choice of h.*

Conjecture 2.22. *The set of cyclotomic units of R is equal to $R^\times$ (see [CDPR16] for a definition of cyclotomic units and a discussion of this conjecture).*

2.4 Representing elements and computing with them

In this section we deal with questions like: how to represent an element of the ring R or of the field K? What is the bit size needed? How do we efficiently compute the Gram-Schmidt orthogonalization of a matrix in K and handle elements in $K_{\mathbb{R}}$?

2.4.1 Computing over rings

In this section and in all the thesis, we assume that we know a LLL-reduced $\mathbb{Z}$-basis $(r_1, \ldots, r_n)$ of $\sigma(R)$ (i.e., R seen as a lattice via the canonical embedding), with respect to $\|\cdot\|$. Note that computing a $\mathbb{Z}$-basis of R is, in the worst-case, an expensive task (see, e.g., [Coh95, Se. 6.1]). Once such a basis is known, applying the LLL algorithm to it has a bit-complexity that is polynomial in $\log \Delta$ and $\max \log \|r_i\|$. Note that the spanned lattice and the positive definite quadratic form may not be integral, but LLL-reduction can be performed by taking approximations to a polynomially bounded precision, because a lower bound for $\lambda_1(R)$ is known (we have $\lambda_1(R) \geq 1$). We refer to [Buc94, SMSV14] for LLL-reduction of non-integral lattices.

Representing elements and ideals. For computations, elements of R can be represented as integer linear combinations of such a LLL-reduced $\mathbb{Z}$-basis of R. The following lemma provides a bound for the involved integer coefficients.

Lemma 2.23 ([FS10, Le. 2]). *Let $(r_i)_{i\leq n}$ be a $\mathbb{Z}$-basis of R that is* LLL*-reduced with respect to $\|\cdot\|$. For all $x = \sum x_i r_i \in K$, we have $\max_i |x_i| \leq 2^{3n/2}\|x\|$.*

We will also use the fact that $\max_i \|r_i\| \leq (4n)^{n/2}\Delta^{1/2}$, which is implied by the facts that $\max_i \|r_i\|$ is no more than 2^n times longer than the last minimum of R (by LLL-reducedness), by Minkowski's second theorem, and the lower bound $\lambda_1(R) \geq 1$.

An ideal can be represented by a $\mathbb{Z}$-basis $(b_i)_{i\leq n}$ with the b_i's belonging to K. The following lemma can be used in combination with Lemma 2.23 to bound the bit-size of a representation of an ideal. The proof follows from the fact that $\det \sigma(I) = \sqrt{\Delta} \cdot \mathcal{N}(I)$ and from standard LLL-reduction inequalities [LLL82, p. 518].

Lemma 2.24. *Let $(b_i)_{i\leq n}$ be a $\mathbb{Z}$-basis of a fractional ideal $I \subset K$ that is* LLL*-reduced with respect to $\|\cdot\|$. Then $\prod_i \|b_i\| \leq 2^{n^2}\sqrt{\Delta} \cdot \mathcal{N}(I)$.*

This lemma implies that an ideal can be represented in size polynomial in $\log \Delta, \log \mathcal{N}(xI)$, and $\log x$ where x is the smallest positive integer such that $xI \subseteq R$.

2.4.2 Computing Gram-Schmidt orthogonalizations

In the LLL algorithm for module lattices of Chapter 4, we will mostly rely on QR-factorization. It carries the same information as Gram-Schmidt orthogonalization, but allows for simpler explanations. However, from a computational perspective, the R-factor may be difficult to represent exactly even for modules contained in K^s, because of the square roots appearing in its definition. This difficulty is circumvented by computing the Gram-Schmidt orthogonalization instead, and using it as a means to represent the R-factor. In this section, we explain how to efficiently compute Gram-Schmidt orthogonalizations.

We first note that the Gram-Schmidt coefficients may not belong to K even if the pseudo-basis does. To explain how to exactly represent the Gram-Schmidt orthogonalization, we need to backtrack a little to operations in K. As seen before (in Lemma 2.23), an element x in R is represented by a vector in $\mathbb{Z}^n$ storing the coefficients of x with respect to a LLL-reduced basis $(r_i)_{i\leq n}$ of R (for $\|\cdot\|$). Multiplication between $x_1, x_2 \in R$ is performed thanks to a table (of $O(n^3)$ integers) storing the representations of each term $r_i r_j$ for all $i, j \leq n$. An element x in K is represented by a pair $(x_{num}, x_{den}) \in R^2$ such that $x = x_{num}/x_{den}$ (and both x_{num} and x_{den} are themselves represented by vectors on $\mathbb{Z}^m$, as explained above). All the above enables additions, multiplications and divisions in K. Now, when computing the Gram-Schmidt orthogonalization, we will make use of complex conjugation in $K_{\mathbb{R}}$ (as we use a Hermitian inner product). Recall that for $x \in K_{\mathbb{R}}$, the element $\bar{x} \in K_{\mathbb{R}}$ is obtained by complex conjugation of its embedding vector. We define $\overline{R}$ and $\overline{K}$ as the subsets of $K_{\mathbb{R}}$ obtained by applying this

operator to the elements of R and K, respectively. These elements can be represented using the $\bar{r}_i$'s rather than the r_i's. We also define $\overline{R}R = \{\bar{y}x : \bar{y} \in \overline{R}, x \in R\}$. Every element $x \in \overline{R}R$ can be expressed as an integer combination of the n^2 elements $\bar{r}_i r_j$ (for $i, j \leq n$), and this vector in $\mathbb{Z}^{n^2}$ is used to represent x. The bit-size of an element in $\overline{R}R$ is the bitsize of this vector in $\mathbb{Z}^{n^2}$. The multiplication table for R allows to perform multiplication in $\overline{R}R$.

Lemma 2.25. *Let $\mathbf{b}_1, \ldots, \mathbf{b}_m \in K^s$ be K-linearly independent. Then the coefficients of the $\mathbf{b}_i^*$'s and μ_{ij}'s can be written as fractions of elements in $\overline{R}R$ whose bit-sizes are polynomially bounded with respect to $\max_i(\log x + \log \|x\mathbf{b}_i\|)$, where x is the smallest positive integer such that $x\mathbf{b}_i \in R^s$ for all i.*

Proof. Without loss of generality, we assume that the $\mathbf{b}_i$'s belong to R^s. Let $d_i = \det(\overline{\mathbf{B}}_i^T \mathbf{B}_i)$ with $\mathbf{B}_i = (\mathbf{b}_1, \ldots, \mathbf{b}_i)$ for $i \leq m$. Then, by a direct adaptation of [LLL82, p. 523], we have $d_i \in \overline{R}R$, $d_{i-1}\mathbf{b}_i^* \in (\overline{R}R)^s$ and $d_j \mu_{ij} \in \overline{R}R$ for all $j < i$. This implies, using Lemma 2.23, that the coefficients of the $\mathbf{b}_i^*$'s and the μ_{ij}'s can be written as fractions of elements in $\overline{R}R$ with d_{i-1} and d_j as denominator, respectively. Going down to the expressions in terms of integer combinations in the r_i's, $\bar{r}_j$'s and $\bar{r}_j r_i$'s, it may be checked that these numerators and denominators are sums and products of a polynomial number of terms $x_i r_i$ and $x_j \bar{r}_j$, where each x_i and x_j is an integer. Further, by Lemma 2.23, each such integer is polynomially bounded with respect to $\max_i \log \|\mathbf{b}_i\|$. This allows to complete the proof.

□

2.5 Probabilities

We let $\Pr[E]$ denote the probability of an event E. For a random variable $x \in \mathbb{R}$, we write $\mathbb{E}[x]$ its expectation and $\mathbb{V}[x]$ its variance. If D is a probability distribution, the notation $x \leftarrow D$ means that x is a random variable sampled according to the distribution D.

2.5.1 Statistics

In Chapter 5, we will do some statistics on random variables defined over the ring $K_\mathbb{R}$ for a power-of-two cyclotomic number field. Below, we start by defining the expectation and variance of such variables, and we give some of their properties. In this subsection, the field K is always a power-of-two cyclotomic field. Let $x = \sum_{1 \leq i < n} x_i X^i$ be a random variable over $K_\mathbb{R}$ (i.e., the x_i's are random variables over $\mathbb{R}$). The expectation of x is defined by $\mathbb{E}[x] = \sum_{1 \leq i < n} \mathbb{E}[x_i] X^i \in K_\mathbb{R}$, and its variance is $\mathbb{V}[x] = \mathbb{E}[x\bar{x}] - \mathbb{E}[x]\mathbb{E}[\bar{x}] \in K_\mathbb{R}$. Note that $\mathbb{V}[x] \in K_\mathbb{R}^+$ for any random variable x over $K_\mathbb{R}$. A random variable x is said centered if $\mathbb{E}[x] = 0$, and isotropic if $\mathbb{V}[x] \sim 1$ (recall that for invertible elements $a, b \in K_\mathbb{R}$, the equivalence $a \sim b$ means that there exists $\alpha \in \mathbb{R}_{>0}$ such that $a = \alpha b$). We recall Hoeffding's inequality.

Theorem 2.26 (Hoeffding's inequality)**.** *Let $Y_1, \cdots, Y_m$ be independent random variables in $\mathbb{R}$ with the same mean $\mu \in \mathbb{R}$ and such that $|Y_i| \leq B$ for all i's. Then for all $t > 0$,*

$$Pr\left[\left|\frac{1}{m}\sum_{i=1}^{m} Y_i - \mu\right| \geq t\right] < 2e^{-\frac{mt^2}{2B^2}}.$$

Hoeffding's inequality, as given above, applies to random variables in $\mathbb{R}$. To use it for our ring $K_\mathbb{R}$, we will view our elements as vectors in $\mathbb{R}^n$ via the coefficient embedding and apply Hoeffding's inequality coefficient-wise.

Corollary 2.27 (Hoeffding's inequality in R)**.** *Let $Y_1, \cdots, Y_m$ be independent random variables in R with the same mean $\mu \in K_\mathbb{R}$ and such that $\|Y_i\|_\infty \leq B$ for all i's. Let $\varepsilon > 0$, then*

$$Pr\left[\left\|\frac{1}{m}\sum_{i=1}^{m} Y_i - \mu\right\|_\infty \geq B\sqrt{\frac{2(\ln n - \ln \varepsilon)}{m}}\right] < 2\varepsilon.$$

Proof. For $1 \leq i \leq m$ and $0 \leq j \leq n-1$, define $Y_{i,j}$ to be the j-th coefficient of the variable $Y_i \in R$ and μ_j to be the j-th coefficient of μ. For a fixed j, the variables $Y_{i,j}$ (where only i varies) are

independent random variables in $\mathbb{R}$ of mean μ_j. Moreover, as $\|Y_i\|_\infty \leq B$ for all i's, the coefficients $Y_{i,j}$ are also bounded by B. We can then apply Hoeffding's inequality (Theorem 2.26) to them. We obtain

$$\begin{aligned}
&\Pr\left[\left\|\frac{1}{m}\sum_{i=1}^{m} Y_i - \mu\right\|_\infty \geq B\sqrt{\frac{2(\ln n - \ln \varepsilon)}{m}}\right] \\
&= \Pr\left[\exists j : \left|\frac{1}{m}\sum_{i=1}^{m} Y_{i,j} - \mu_j\right| \geq B\sqrt{\frac{2(\ln n - \ln \varepsilon)}{m}}\right] \\
&\leq \sum_{j=0}^{n-1} \Pr\left[\left|\frac{1}{m}\sum_{i=1}^{m} Y_{i,j} - \mu_j\right| \geq B\sqrt{\frac{2(\ln n - \ln \varepsilon)}{m}}\right] \\
&< \sum_{j=0}^{n-1} 2e^{-\frac{2mB^2(\ln n - \ln \varepsilon)}{2B^2 m}} = \sum_{j=0}^{n-1} 2 \cdot \frac{\varepsilon}{n} = 2\varepsilon.
\end{aligned}$$

We used the union bound and Hoeffding's inequality with $t = B\sqrt{\frac{2(\ln n - \ln \varepsilon)}{m}}$. □

2.5.2 Discrete Gaussians

For any real $\sigma > 0$ and point $c \in \mathbb{R}^n$, the (spherical) Gaussian weight function is defined over $\mathbb{R}^n$ by

$$\rho_{\sigma,c}(x) = \exp\left(\frac{-\|x-c\|^2}{2\sigma^2}\right).$$

For any lattice $L \subset \mathbb{R}^n$, we define the discrete (spherical) Gaussian distribution over L of parameter σ and centered in c by

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

where $\rho_{\sigma,c}(L) = \sum_{x\in L} \rho_{\sigma,c}(x)$. We simplify $\rho_{\sigma,0}$ and $D_{L,\sigma,0}$ into ρ_σ and $D_{L,\sigma}$, and say in that case that the distribution is centered.

For any lattice $L \subset \mathbb{R}^n$ and $\varepsilon > 0$, the smoothing parameter of L is defined as

$$\eta_\varepsilon(L) = \min\{s > 0 : \rho_{1/s}(L^*) \leq \varepsilon\},$$

where $L^* = \{w : \langle x, w\rangle \in \mathbb{Z}, \forall x \in L\}$ is the dual lattice of L. Lemma 3.3 of [MR07] states that for any function $\omega(\log n)$, there exists a negligible ε such that $\eta_\varepsilon(L) \leq \sqrt{\omega(\log n)} \cdot \lambda_n(L)$. Further, Lemma 4.4 of [MR07] also states that for any parameter $\sigma \geq \eta_\varepsilon$ (for some $0 < \varepsilon < 1$), we have the following tail bound

$$\Pr_{x \leftarrow D_{L,\sigma,c}}(\|x-c\| > \sigma\sqrt{n}) \leq \frac{1+\varepsilon}{1-\varepsilon} 2^{-n}. \tag{2.15}$$

Non-spherical Gaussian distributions. In Chapter 5, we will use non-spherical Gaussian distributions over $K_\mathbb{R}$, with K a power-of-two cyclotomic field, and using the coefficient embedding. Compared to the spherical Gaussian distributions described above, these non-spherical distributions are parametrized by a parameter $\Sigma \in K_\mathbb{R}^+$, instead of $\sigma^2 \in \mathbb{R}$. If we choose $\Sigma \in \mathbb{R}_{>0} \subset K_\mathbb{R}^+$, then we recover the spherical Gaussian distribution defined above, hence, we keep the same notations as for spherical Gaussians. We will consider these distributions on lattices of the form $I + y$ for I a principal fractional ideal of K and $y \in K_\mathbb{R}$. For $\Sigma \in K_\mathbb{R}^+$ and $c \in K_\mathbb{R}$, we define the (non-spherical) Gaussian weight function on $K_\mathbb{R}$ as

$$\rho_{\sqrt{\Sigma},c} : x \mapsto \exp\left(-\frac{1}{2}\left\|\frac{x-c}{\sqrt{\Sigma}}\right\|^2\right).$$

For any shifted ideal $I+y$, $I \subset K$, $y \in K_\mathbb{R}$, we define the (non-spherical) discrete Gaussian distribution over $I + y$ of parameter $\sqrt{\Sigma}$, centered in c by:

$$\forall x \in I + y,\ D_{I+y,\sqrt{\Sigma},c}(x) = \frac{\rho_{\sqrt{\Sigma},c}(x)}{\rho_{\sqrt{\Sigma},c}(I+y)}.$$

For concision again, we write $D_{I+y,\sqrt{\Sigma}}$ instead of $D_{I+y,\sqrt{\Sigma},0}$ and $\rho_{\sqrt{\Sigma}}$ instead of $\rho_{\sqrt{\Sigma},0}$.

We know that for a principal ideal gR of $K_{\mathbb{R}}$, we have $\lambda_n(gR) \leq \|g\|$, because the $X^i g$ form a set of n linearly independent vectors of gR of Euclidean norm $\|g\|$. Hence, we obtain that for any function $\omega(\log n)$, there exists a negligible ε such that

$$\eta_\varepsilon(gR) \leq \sqrt{\omega(\log n)}\|g\|.$$

Let $g \in K$. If $\sqrt{\Sigma} = \sigma \in \mathbb{R}$ (i.e., $\sqrt{\Sigma} \in \mathbb{R}[X]/(X^n+1)$ has a representative of degree 0), is such that $\|g/\sqrt{\Sigma}\| = o(1/\sqrt{\log n})$, then we have $\sigma = \sqrt{\Sigma} = \sqrt{\omega(\log n)}\|g\|$ and the tail bound inequality (2.15) gives us

$$\Pr_{x \leftarrow D_{gR,\sigma,c}}(\|x-c\| > \sigma\sqrt{n}) \leq 2 \cdot 2^{-n}. \tag{2.16}$$

For an arbitrary $\Sigma \in K_{\mathbb{R}}^+$, the distribution $D_{I+y,\sqrt{\Sigma},c}$ is the same as $(\sqrt{\Sigma} \cdot D_{I/\sqrt{\Sigma},1,(c-y)/\sqrt{\Sigma}} + y)$ (i.e. sampling $x \leftarrow D_{I+y,\sqrt{\Sigma},c}$ is the same as sampling $x' \leftarrow D_{I/\sqrt{\Sigma},1,(c-y)/\sqrt{\Sigma}}$ and outputting $x = \sqrt{\Sigma} \cdot x' + y$). Hence, we can always transform a non-spherical Gaussian distribution into a spherical one (with a real parameter $\sqrt{\Sigma}$), and a non shifted ideal. We can then apply the previous tail bound for spherical Gaussian distributions. If $I = gR$ is a principal ideal and $\|g/\sqrt{\Sigma}\| = o(1/\sqrt{\log n})$, then we have

$$\Pr_{x \leftarrow D_{gR+y,\sqrt{\Sigma},c}}(\|x-c\| > n \cdot \|\sqrt{\Sigma}\|) \leq 2 \cdot 2^{-n}. \tag{2.17}$$

Indeed, let $x' \leftarrow D_{I/\sqrt{\Sigma},1,(c-y)/\sqrt{\Sigma}}$ and $x = \sqrt{\Sigma} \cdot x' + y$ (recall that the distribution of x is $D_{gR+y,\sqrt{\Sigma},c}$). Then we have

$$\|x-c\| = \|\sqrt{\Sigma} \cdot (x' - (c-y)/\sqrt{\Sigma})\| \leq \sqrt{n} \cdot \|\sqrt{\Sigma}\| \cdot \|(x' - (c-y)/\sqrt{\Sigma})\|.$$

We conclude by applying the tail bound inequality (2.16) to the distribution $D_{I/\sqrt{\Sigma},1,(c-y)/\sqrt{\Sigma}}$ to obtain $\|(x' - (c-y)/\sqrt{\Sigma})\| \leq \sqrt{n}$ with probability at least $1 - 2 \cdot 2^{-n}$.

The next lemma states that if the standard deviation is larger than the smoothing parameter (recall that this is what the condition $\|g/\sqrt{\Sigma}\| = o(1/\sqrt{\log n})$ ensures), then one can efficiently sample from a distribution negligibly close to a non-spherical Gaussian distribution.

Theorem 2.28 (Reformulation of Theorem 4.1 of [GPV08])**.** *There exists a probabilistic polynomial time algorithm that given $g \in R$, $y \in K_{\mathbb{R}}$ and a parameter $\Sigma \in K_{\mathbb{R}}^+$ such that $\|g/\sqrt{\Sigma}\| \leq o(1/\sqrt{\log n})$, outputs x from a distribution negligibly close to $D_{gR+y,\sqrt{\Sigma},c}$.*

The following lemma states that, when the standard deviation is larger than the smoothing parameter, a discrete Gaussian resembles the continuous Gaussian, in particular it is almost centered at c, and of variance almost Σ. In the following, because these properties, we sometimes abuse notation and call Σ the variance parameter of the Gaussian distribution.

Lemma 2.29 (Adapted from Lemma 4.2 of [MR07])**.** *For any $g \in K$, $\Sigma \in K_{\mathbb{R}}^+$ and $c, y \in K_{\mathbb{R}}$ such that $\|g/\sqrt{\Sigma}\| \leq o(1/\sqrt{\log n})$, if $x \leftarrow D_{gR+y,\sqrt{\Sigma},c}$, then $\|\mathbb{E}[x] - c\| \leq \varepsilon \cdot \|\sqrt{\Sigma}\|$ and $\|\mathbb{V}[x] - \Sigma\| \leq \varepsilon \cdot \|\Sigma\|$ for some negligible function $\varepsilon(n)$.*

2.6 Matrix branching programs

We recall in this section the definition of matrix branching programs, and we introduce some notation that will be used in Chapter 6 of this thesis. A branching program is defined over a ring $\mathcal{R}$.

Definition 2.30 (d-ary Matrix Branching Program)**.** A d-ary matrix branching program $\mathbf{A}$ of length ℓ and width w over m-bit inputs is given by a sequence of square matrices

$$\{A_{i,\mathbf{b}}\}_{i \in \{1,\ldots,\ell\}, \mathbf{b} \in \{0,1\}^d} \in \mathcal{R}^{w \times w},$$

two bookend vectors

$$A_0 \in \mathcal{R}^{1 \times w} \text{ and } A_{\ell+1} \in \mathcal{R}^{w \times 1},$$

and an input function $\mathtt{inp} : \{1, \ldots, \ell\} \to \{1, \ldots, m\}^d$.

Let $x \in \{0,1\}^m$ and let x_i denote the i-th bit of x, for i in $\{1, \ldots, m\}$. We will use the notation $x[\mathtt{inp}(i)] = (x_{\mathtt{inp}(i)_1}, x_{\mathtt{inp}(i)_2}, \cdots, x_{\mathtt{inp}(i)_d}) \in \{0,1\}^d$, where $\mathtt{inp}(i) = (\mathtt{inp}(i)_1, \cdots, \mathtt{inp}(i)_d) \in \{1, \ldots, m\}^d$.

The output of the matrix branching program on input $x \in \{0,1\}^m$ is given by

$$\mathbf{A}(x) = \begin{cases} 0 & \text{if } A_0 \cdot \left(\prod_{1 \le i \le \ell} A_{i,x[\mathtt{inp}(i)]}\right) \cdot A_{\ell+1} = 0 \\ 1 & \text{otherwise.} \end{cases}$$

We will write the branching programs in capital bold letters, and their matrices in capital letters (but not bold). In this thesis, we will only consider matrix branching programs (and no general branching programs). For readability reason, we will often forget the term "matrix" and simply call them branching programs (or BP for short). A branching program with $d = 1$ (respectively with $d = 2$) is also called a single input (respectively dual input) branching program. In the following, we will not distinguish between the single input and dual input cases, as the attack presented in Chapter 6 works in the same way in both cases (and even for higher arity d).

We say that two branching programs are equivalent if they compute the same function. We also introduce a notion of strong equivalence between branching programs.

Definition 2.31 (Strongly equivalent branching programs)**.** We say that two d-ary matrix branching programs $\mathbf{A} = (A_0, \{A_{i,\mathbf{b}}\}_{1 \le i \le \ell, \mathbf{b} \in \{0,1\}^d}, A_{\ell+1})$ and $\mathbf{A'} = (A'_0, \{A'_{i,\mathbf{b}}\}_{1 \le i \le \ell, \mathbf{b} \in \{0,1\}^d}, A'_{\ell+1})$, with the same length ℓ and the same input function $\mathtt{inp}$ (but not necessarily defined over the same rings) are strongly equivalent if, for all $\{\mathbf{b}_i\}_{1 \le i \le \ell} \in (\{0,1\}^d)^\ell$, we have

$$A_0 \cdot \prod_{1 \le i \le \ell} A_{i,\mathbf{b}_i} \cdot A_{\ell+1} = 0 \iff A'_0 \cdot \prod_{1 \le i \le \ell} A'_{i,\mathbf{b}_i} \cdot A'_{\ell+1} = 0. \tag{2.18}$$

This notion is stronger than simple equivalence between branching programs, because we ask that (2.18) holds for all possible choices of $\{\mathbf{b}_i\}_{1 \le i \le \ell}$, and not only for the ones of the form $\{x[\mathtt{inp}(i)]\}_i$ for some input x (corresponding to an honest evaluation of the branching program on x). The pair of branching programs described in Section 6.3.4 gives an example of two equivalent branching programs that are not strongly equivalent.

Chapter 3

SVP in Ideal Lattices with Pre-Processing

Finding short vectors in a lattice is a fundamental problem in lattice-based cryptography. In this chapter, we are interested in finding somehow short vectors in *ideal* lattices. Our objective is not to find the shortest non-zero vector of a given ideal, but rather to obtain better trade-offs between time and approximation factor than what is currently known for arbitrary lattices (i.e., using the BKZ algorithm [SE94]), by allowing some pre-computation. We will hence be interested in finding approximations of the shortest non-zero vector of a given ideal, for approximation factors 2^{n^α}, where n is the dimension of the lattice and $\alpha \in (0, 1]$.

In a first section, we explain how the problem of finding short vectors in ideal lattices can be transformed into a closest vector problem instance in a fixed lattice, depending only on the number field. This transformation is formalized in Theorem 3.5 and does not require any heuristic argument, except for the generalized Riemann hypothesis. In Section 3.4, we explain how to solve the CVP instance in the fixed lattice using Laarhoven's algorithm (see Theorem 2.6), which has a pre-processing phase whose run-time is exponential in the dimension of the lattice. In order to use Laarhoven's algorithm, we have to introduce some new heuristic assumptions. Finally, we conclude by instantiating Theorem 3.5 with Laarhoven's algorithm and the currently best known algorithms to solve class-group related problems in arbitrary number fields. We also give an alternative instantiation of Theorem 3.5, replacing Laarhoven's algorithm by an oracle solving CVP in the fixed lattice. This alternative instantiation will be used in Chapter 4.

This chapter corresponds to a joint work with Guillaume Hanrot and Damien Stehlé, which was published in the proceedings of Eurocrypt 2019 [PHS19]. The code that was used to perform the experiments described in this chapter is available at

`http://perso.ens-lyon.fr/alice.pellet___mary/code/code-approx-ideal-svp.zip`

Contents

3.1 Introduction

The Learning With Errors problem (LWE) introduced by Regev in [Reg05] has proved invaluable towards designing cryptographic primitives. However, as its instance bit-sizes grow at least quadratically with the security parameter to be well-defined, LWE often results in primitives that are not very efficient. In order to improve the efficiency, Stehlé, Steinfeld, Tanaka and Xagawa [SSTX09] introduced the search Ideal-LWE problem which involves polynomials modulo X^n+1 for n a power of two, and Lyubashevsky, Peikert and Regev [LPR10] exhibited the relationship to power-of-two cyclotomic fields, gave a reduction from the latter search problem to a decision variant, and tackled more general rings. This is now referred to as Ring-LWE, and leads to more efficient cryptographic constructions. To support the conjecture that Ring-LWE is computationally intractable, the authors of [SSTX09, LPR10] gave polynomial-time quantum reductions from the approximate Shortest Vector Problem (approx-SVP) restricted to ideal lattices to Ring-LWE. Approx-SVP consists in finding a non-zero vector of an input lattice, whose norm is within a prescribed factor from the lattice minimum. Ideal lattices are lattices corresponding to ideals of the ring of integers of a number field, for example a power-of-two cyclotomic field in the situation above. When considering a lattice problem for such an ideal, the ideal is implicitly viewed as a lattice via the canonical embedding. A third quantum reduction from approx-SVP for ideal lattices to Ring-LWE was proposed by Peikert, Regev and Stephens-Davidowitz [PRS17]. It has the advantage of working for all number fields.

As is always the case, the value of these reductions highly depends on the intractability of the starting problem, i.e., approx-SVP for ideal lattices: approx-SVP for ideal lattices could even turn out to be computationally easy to solve, hence making these reductions vacuous. We stress that even if this were the case, that would not necessarily mean that there exists an efficient algorithm for Ring-LWE. In this chapter, we investigate the intractability of ideal approx-SVP for arbitrary number fields.

For arbitrary lattices, the best known trade-off between the run-time and the approximation factor is given by Schnorr's hierarchy of reduction algorithms [Sch87], whose most popular variant is the BKZ algorithm [SE94]. For any real number $\alpha \in [0,1]$ and any lattice L of dimension n given by an arbitrary basis, it allows one to compute a vector of $L \setminus \{0\}$ which is no more than $2^{\widetilde{O}(n^\alpha)}$ times longer than a shortest one, in time $2^{\widetilde{O}(n^{1-\alpha})}$ (assuming the bit-size of the input basis is polynomial in n). This trade-off is drawn in blue in Figure 3.1.[1] In the case of ideal lattices in a cyclotomic ring of prime-power conductor (i.e., the ring of integers of $\mathbb{Q}(\zeta_m)$ where m is a prime power and ζ_m is a complex primitive m-th root of unity), it has been shown that it is possible to obtain a better trade-off than the BKZ algorithm, in the quantum computation setting. For *principal* ideal lattices, i.e., ideals that can be generated by a single element, the algorithmic blueprint, described in [CGS14b, Ber14], consists in first using class group computations to find a generator of the ideal, and then use the so-called log-unit lattice to shorten the latter generator (we note that using the log-unit lattice for this purpose was already suggested in [RBV04]). A quantum polynomial-time algorithm for the first step was provided by Biasse and Song [BS16], building upon the work of [EHKS14]. The second step was carefully analyzed by Cramer, Ducas, Peikert and Regev [CDPR16], resulting in a quantum polynomial-time algorithm for approx-SVP restricted to *principal* ideal lattices, with a $2^{\widetilde{O}(\sqrt{n})}$ approximation factor. (See [HWB17] for a generalization to cyclotomics with degree of the form $p^\alpha q^\beta$, with p and q prime.) This line of work was extended by Cramer, Ducas and Wesolowski [CDW17] to any (not necessarily principal) ideal lattice of a cyclotomic ring of prime-power conductor. Put together, these results give us the trade-off between approximation factor and run-time drawn in red dashes in Figure 3.1. This is better than the BKZ algorithm when the approximation factor is larger than $2^{\widetilde{O}(\sqrt{n})}$. However, for smaller approximation factors, Schnorr's hierarchy remains the record holder. One could also hope to improve the trade-off for classical computing, by replacing the quantum principal ideal solver of [BS16] by the classical one of Biasse, Espitau, Fouque, Gélin and Kirchner [BEF+17]. However, this classical principal ideal solver runs in sub-exponential time $2^{\widetilde{O}(\sqrt{n})}$, hence combining it with [CDPR16, CDW17] results in a classical approx-SVP algorithm for a $2^{\widetilde{O}(\sqrt{n})}$ approximation factor in time $2^{\widetilde{O}(\sqrt{n})}$. Up to the $\widetilde{O}(\cdot)$ terms, this is exactly the trade-off obtained using Schnorr's hierarchy. Recently, Ducas, Plançon and Wesolowski [DPW19] experimentally analysed the $\widetilde{O}(\cdot)$ term of the $2^{\widetilde{O}(\sqrt{n})}$ approximation factor of the [CDPR16, CDW17] algorithm. This allows them to determine for which dimension n this quantum

[1] This figure, like all similar ones in this chapter, is in $(\log_n \log_2)$-scale for both axes.

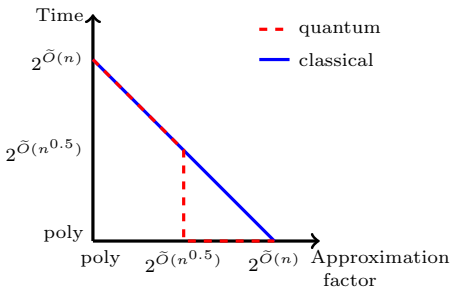

Figure 3.1: Prior time/approximation trade-offs for ideal approx-SVP in cyclotomic fields of prime-power conductor.

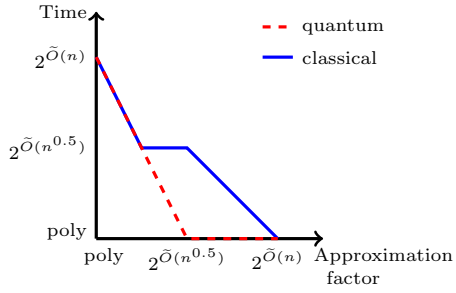

Figure 3.2: New trade-offs for ideal approx-SVP in the same fields (with a pre-processing of cost $\exp(\widetilde{O}(n))$).

algorithm outperforms BKZ.

3.2 Contribution

It is a classical fact due to Minkowski [Min67, pp. 261–264] that there exists an absolute constant $c > 1$ such that for all number fields K of degree $n \geq 2$ and discriminant Δ, we have $|\Delta| > c^n$. In the sequel, we shall thus state all our upper bounds in terms of $\log|\Delta| \geq \Omega(n)$. Actually, to fix the ideas, one may consider $\log|\Delta| = \widetilde{O}(n)$, which is the case for cyclotomic fields.

Let us consider a number field K of degree n and discriminant Δ. We assume a basis of the ring of integers R of K is given. Our algorithm performs some pre-processing on K, in exponential time $2^{\widetilde{O}(\log|\Delta|)}$. Once this pre-processing phase is completed and for any $\alpha \in [0, 1/2]$, the algorithm can, given any ideal lattice I of R, output a $2^{\widetilde{O}((\log|\Delta|)^{\alpha+1}/n)}$ approximation of a shortest non-zero vector of I in time $2^{\widetilde{O}((\log|\Delta|)^{1-2\alpha})} + T_{\text{c-g}}(K)$. Here $T_{\text{c-g}}(K)$ denotes the time needed to perform class group related computations in K: computing relations between elements of the class group and computing the units of R. Using the results of [BS16, BEF+17, BF14], we can replace $T_{\text{c-g}}(K)$ by $\text{poly}(\log|\Delta|)$ for a quantum computer, and, for a classical computer, by $2^{\widetilde{O}((\log|\Delta|)^{1/2})}$ if K is a cyclotomic field of prime-power conductor and by $2^{\widetilde{O}((\log|\Delta|)^{2/3})}$ for an arbitrary field K. The three algorithms rely on the Generalized Riemann Hypothesis (GRH) and the two sub-exponential algorithms in the classical setting also require additional heuristic assumptions. The correctness and cost analyses of our algorithm rely on these heuristic assumptions, and others. Our contribution is formalized in the theorem below, which is the main result of this chapter.

Theorem 3.1 (Heuristic, see Theorems 3.5 and 3.10)**.** *Let $\alpha \in [0, 1/2]$ and K be a number field of degree n and discriminant Δ. Assume that a basis of the ring of integers R of K is known. Under some conjectures and heuristics, there exist two algorithms $A_{\text{pre-proc}}$ and A_{query} such that*

- *Algorithm $A_{\text{pre-proc}}$ takes as input the ring R, runs in time $2^{\widetilde{O}(\log|\Delta|)}$ and outputs a hint w of bit-size $2^{\widetilde{O}((\log|\Delta|)^{1-2\alpha})}$;*
- *Algorithm A_{query} takes as inputs any ideal I of R (whose algebraic norm has bit-size bounded by $2^{\text{poly}(\log|\Delta|)}$) and the hint w output by $A_{\text{pre-proc}}$, runs in time $2^{\widetilde{O}((\log|\Delta|)^{1-2\alpha})} + T_{\text{c-g}}(K)$, and outputs an element $x \in I$ such that $0 < \|x\|_2 \leq 2^{\widetilde{O}((\log|\Delta|)^{\alpha+1}/n)} \cdot \lambda_1(I)$.*

The hint output by the pre-processing phase has a bit-size that is bounded by the run-time of the query phase. By considering larger hints, the run-time of the query phase could be drastically improved. We give more details below, at the end of the high-level description of the algorithm.

Considering only the query cost, this result is of interest when $\log|\Delta| \leq \widetilde{O}(n^{4/3})$ for quantum computations and $\log|\Delta| \leq \widetilde{O}(n^{12/11})$ for classical computations. Indeed, in the other cases, the time/quality trade-offs obtained by our algorithm are worse than the ones obtained using Schnorr's hierarchy of algorithms. By letting α vary in $[0, 1/2]$ and considering cyclotomic fields of prime-power

conductor, we obtain the trade-offs represented in Figure 3.2. For a discussion for more general values of $\log|\Delta|$, we refer to Section 3.5. Going back to cyclotomic fields of prime-power conductor, these new trade-offs improve upon the prior ones, both for quantum and classical computers. Note that in Figure 3.2, we only plot the time needed for the query phase of the algorithm, but there is a pre-processing phase of exponential time performed before. Also, the new algorithm is no better than Schnorr's hierarchy in the classical setting when the run-time is sufficiently small. Hence, in Figure 3.2, we plotted the trade-offs obtained using Schnorr's hierarchy when they are better than the ones obtained with the new algorithm. The query phase of the new algorithm gives a quantum acceleration for approx-SVP for ideal lattices in cyclotomic fields of prime-power conductor, for all approximation factors $2^{\widetilde{O}(n^\alpha)}$ with $\alpha \in (0,1)$. This extends [CDW17], which obtained such a quantum acceleration for $\alpha \in [1/2,1)$. The query phase of the new algorithm also gives a classical acceleration for these fields, but only for $\alpha \in (0,1/2)$.

3.2.1 Technical overview

Our algorithm is inspired by the algorithms in [CDPR16, CDW17]. Given an ideal I as input, the idea is to first find a principal ideal J contained in I (using [CDW17]), and then compute a short generator of this ideal J (using [CDPR16]). This short generator is a somehow small element of I. This approach provides a $2^{\widetilde{O}(\sqrt{n})}$ approximation factor for approx-SVP in I. However, it can be shown that we cannot improve this approximation factor using these techniques, even if we increase the run-time of the algorithm. The reason is that, given an arbitrary principal ideal J, it may be that its shortest generator is $2^{\widetilde{O}(\sqrt{n})}$ times longer than its shortest non-zero vector.

We modify the strategy above, as follows. Given any ideal I, we try to find a 'good' principal ideal J contained in I, where we say that a principal ideal is 'good' if its shortest generator is not much larger than its shortest non-zero vector. The precise definition of 'not much larger' will depend on the approximation factor we want to achieve for our approx-SVP instance. Because the Euclidean norm of the shortest non-zero vector of J (broadly) increases with its algebraic norm, we also require that the algebraic norm of J is not much larger than the one of I (note that this was already needed in [CDPR16, CDW17]). To find this 'good' principal ideal J, the main idea of our algorithm is to express the problem as a Closest Vector Problem (CVP) instance in a lattice L depending only on the number field K.

This lattice L is similar to the one appearing in sub-exponential algorithms for computing the class group (see for instance [HM89, Buc88]). More precisely, we first select a set $\mathfrak{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_r\}$ of prime ideals of polynomially bounded algebraic norms, generating the class group. We then compute a generating set of the $\mathfrak{B}$-units, i.e., the set of elements $u \in K$ for which there exists $(e_1, \ldots, e_r) \in \mathbb{Z}^r$ such that $\langle u \rangle = \prod_i \mathfrak{p}_i^{e_i}$. The lattice L is obtained by considering the integer linear combinations of vectors of the form $(\operatorname{Log} u, e_1, \ldots, e_r)^T$, where $\langle u \rangle = \prod_i \mathfrak{p}_i^{e_i}$ and Log is the map applying the logarithm function to the canonical embedding, coefficient-wise. This lattice L only depends on the field K and can then be pre-computed and pre-processed.

Given any ideal I, the query phase of our algorithm computes a target vector t from I, and then solves a CVP instance in L with this target vector t. First, we decompose the ideal I in the class group as a product of the ideals of $\mathfrak{B}$. Concretely, we compute $g \in K$ and $(v_1, \ldots, v_r) \in \mathbb{Z}^r$ such that $I = \prod_i \mathfrak{p}_i^{v_i} \cdot \langle g \rangle$. This principal ideal $\langle g \rangle$ is a candidate for our principal ideal J contained in I (assume for the moment that the v_i's are non-positive, so that $\langle g \rangle$ is indeed contained in I). However, as is, we have no guarantee that $\langle g \rangle$ has a short generator. We also have no guarantee that its algebraic norm is not much larger than the one of I (i.e., that the v_i's are small). Hence, our objective is to multiply the principal ideal $\langle g \rangle$ by other principal ideals, until we have a good candidate for J. To do so, we define the vector $t = (-\operatorname{Log} g, v_1, \ldots, v_r)^T$. Observe that $\langle g \rangle$ would be a good candidate for J if this vector was short (and with $v_i \leq 0$ for all i). Indeed, this would mean that g is a short generator of $\langle g \rangle$ (because $\operatorname{Log} g$ is short), and that $\langle g \rangle = I \cdot \prod_i \mathfrak{p}_i^{-v_i}$ is a small multiple of I (because the $\mathfrak{p}_i$'s have polynomially bounded norms, and the v_i's are small; the non-positivity of the v_i's is used to ensure that the ideal $\prod_i \mathfrak{p}_i^{-v_i}$ is integral). Also, we can see that adding a vector of L to t amounts to multiplying the principal ideal $\langle g \rangle$ by another principal ideal (corresponding to the vector of L we are adding). Hence, we can find a good candidate J (together with a short generator of J) by solving a CVP instance in L with target t.

Finally, we need to solve CVP in the lattice L. We do not know any basis for L which would enable us to solve CVP in it efficiently (as opposed to the lattices considered in [CDPR16, CDW17]). However, the lattice L is fixed for a given number field, hence we can pre-process it. For this, we use a CVP with pre-processing (CVPP) algorithm due to Laarhoven [Laa16]. This leads to the time/approximation trade-offs given in Theorem 3.1. In [Laa16], significant effort is spent towards minimizing the constant factors in the exponents. These have recently been improved in [DLW19]. In this work, we neglect these factors for the sake of simplicity, but these would clearly matter in practice.

Laarhoven's CVPP algorithm is such that the bit-size of the output of the pre-processing phase is no larger than the run-time of the query phase[2] (hence, it is also the case for our algorithm). If we do not require this, we could have the following very simple and efficient algorithm for CVPP. First, it computes a short basis B_{sh} of the lattice. Then, it partitions the fundamental parallelepiped associated to B_{sh} into exponentially many small boxes, such that given any point of the parallelepiped, it is easy to determine to which box it belongs. Then, for each of these boxes, the pre-processing algorithm would compute a closest point of the lattice. The output of the pre-processing phase would then be the small basis B_{sh} and the set of all boxes together with their closest lattice point. Finally, given any vector in the real span of the lattice, the query algorithm would reduce it modulo B_{sh} to obtain a vector in the fundamental parallelepiped, and then determine the box of this reduced vector and its associated lattice vector. All this can be done efficiently (assuming we can efficiently access the database) and provides a small factor approximation for CVP, at the expense of a huge database.

Overall, the correctness and cost analyses of our algorithm rely on several heuristic assumptions. Many of them come from previous works [Laa16, BEF+17, BF14] and were already analysed. We introduce three new heuristic assumptions: Heuristics 3.6, 3.7 and 3.8 in Section 3.4. We discuss them by providing some mathematical justifications and some experimental results corroborating them. Concurrently to this work, Stephens-Davidowitz [SD19] obtained a provable variant of the CVPP trade-offs from [Laa16, DLW19] that we use. Relying on it would allow us to make do with Heuristic 2.7, which was inherited from [Laa16, DLW19], at the expense of replacing Heuristic 3.6 by a similar one on the smoothing parameter of the lattice under scope (rather than its covering radius).

3.2.2 Impact

The query phase of the new algorithm can be interpreted as a non-uniform algorithm, as it solves approx-SVP for ideals of K, using a hint depending on K only. As the time needed to compute that hint (i.e., the run-time of $A_{\text{pre-proc}}$) is exponential, the concrete impact is limited. Nevertheless, our result should rather be interpreted as a strong indication that ideal approx-SVP is most likely a weaker problem than approx-SVP for arbitrary lattices: for unstructured lattices, there is no known non-uniform algorithm outperforming Schnorr's hierarchy.

Few cryptographic constructions have their security impacted by faster ideal approx-SVP solvers. An important example, related to the topic of this thesis, is the GGH13 cryptographic multilinear map [GGH13a] and its extensions. We will see in Section 6.3.2 that being able to find a somehow short vector in a principal ideal lattice can be used to mount an attack against several candidate obfuscators based on the GGH13 map. Note however that even if the difficulty of ideal-SVP has an impact on the security of the GGH13 map, our algorithm does not provide a concrete attack on this scheme, because of the exponential pre-processing time.

More importantly, our result strongly suggests that approx-SVP for ideals of the ring of integers R of a number field K may be weaker than Ring-LWE, for a vast family of number fields. Up to some limited parameter losses, Ring-LWE and approx-SVP for R-modules over K (with ranks ≥ 2) reduce to one another [LS15, AD17]. Therefore, a separation between approx-SVP for ideals and Ring-LWE is essentially the same as a separation between approx-SVP for ideals and approx-SVP for R-modules over K.

[2] Laarhoven also describes a variant of his algorithm in which he uses locality-sensitive hashing to reduce the run-time of the query phase below the bit-size of the advice, but we are not considering this variant here.

3.3 From Ideal SVP to CVP in a Fixed Lattice

The main idea of our algorithm is, given an input ideal I, to find a principal ideal $\langle g \rangle \subseteq I$ with a short generator g. This is very similar to [CDW17], where the authors find a $2^{O(\sqrt{n})}$ approximation of a shortest non-zero vector of the ideal I by computing a principal ideal contained in I and then finding a short generator of this principal ideal. The limitation of this approach is that, if we consider any principal ideal contained in I, we cannot hope to find a better approximation than the $2^{O(\sqrt{n})}$ approximation obtained above in the worst case. This is due to the fact that in some principal ideals (including for prime-power cyclotomic fields), the shortest generator can be $2^{O(\sqrt{n})}$ times longer than a shortest non-zero element of the ideal (see [CDPR16]). Instead of looking for any principal ideal contained in I, we consider only those with a 'good' generator (i.e., a generator which is also a very short element of the corresponding principal ideal).

In order to find such an ideal, we merge the two steps of [CDPR16, CDW17] (consisting in first finding a principal multiple of I and then computing a small generator of the principal ideal), by introducing a lattice L that is very similar to the one used for class group computations. This lattice only depends on the number field (and not on the ideal I). We describe it in the next subsection. We then show how to express the problem of finding a principal multiple of I with a small generator as a CVP instance for this fixed lattice.

3.3.1 Definition of the lattice L

In this subsection, we define the lattice L which we will use in order to transform our ideal-SVP instance into a CVPP' instance. We also give an algorithm to compute a basis of L and analyze its run-time. The lattice L we are considering is not new. It was already used in previous sub-exponential algorithms computing the class group of a number field [HM89,Buc88,BF14,BEF+17,Gel17]. However, these algorithms usually choose a sub-exponential set of ideals, hence resulting in a lattice L of sub-exponential dimension. Our lattice L will have a dimension which is polynomial in $\log |\Delta|$.

In the following, we fix some integer r such that $\log h_K \leq r$ and $r \leq \text{poly}(\log |\Delta|)$ (looking forward, the integer r will be related to the dimension of the lattice in which we will solve CVPP', so it would be undesirable to set it too large). Let us also fix a set of prime integral ideals $\mathfrak{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_r\}$ as given by Corollary 2.15. We consider the lattice L of dimension $\nu := r + r_1 + r_2 - 1$, generated by the columns of the following matrix:

$$B_L := \begin{array}{c|c} \overbrace{}^{r_1+r_2-1} & \overbrace{\phantom{c \cdot \widetilde{h}_{g_1}, \ldots, c \cdot \widetilde{h}_{g_r}}}^{r} \\ \hline c \cdot B_\Lambda & c \cdot \widetilde{h}_{g_1}, \ldots, c \cdot \widetilde{h}_{g_r} \\ \hline 0 & v_1 \quad v_2 \quad \cdots \quad v_r \end{array} \begin{array}{l} \\ \left.\begin{array}{l} \\ \end{array}\right\} r_1 + r_2 - 1 \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} r \end{array} \quad \left.\begin{array}{l} \\ \\ \end{array}\right\} \nu$$

where:

- the scaling parameter $c > 0$ is to be chosen later;
- the matrix $B_\Lambda = (f_{H \cap E}(b_1), \ldots, f_{H \cap E}(b_{r_1+r_2-1}))$ is a basis of $f_{H \cap E}(\Lambda)$, where Λ is the log-unit lattice and $f_{H \cap E} : H \cap E \subset \mathbb{R}^n \to \mathbb{R}^{r_1+r_2-1}$ is an isometry;[3]

[3] As Λ is not full rank in $\mathbb{R}^n$, we change the ambient space such that $f_{H \cap E}(\Lambda)$ becomes full rank in $H \cap E = \mathbb{R}^{r_1+r_2-1}$. Note however that the ℓ_2-norm is preserved by this transformation (this is not the case for the ℓ_1 and ℓ_∞ norms).

- the matrix consisting of the vectors $v_i = (v_{1i}, \ldots, v_{ri})^T$ is a basis of $\ker(f_{\mathfrak{B}})$, where $f_{\mathfrak{B}}$ is defined in Section 2.3.10 (in particular, the ideals $\prod_j \mathfrak{p}_j^{v_{ji}}$ are principal for all i);
- the column vectors $\widetilde{h}_{g_i}$ are of the form $f_{H\cap E}(\pi_H(\operatorname{Log} g_i))$ for $g_i \in K$ a generator of the fractional principal ideal associated with the relation v_i, i.e., we have $\prod_j \mathfrak{p}_j^{v_{ji}} = \langle g_i \rangle$.

We will explain how to construct L below. This lattice enjoys the following property, which will be used later.

Lemma 3.2. *Let w be a vector of L and parse it as $w = (h, v)^T$ with h of dimension $r_1 + r_2 - 1$ and $v = (v_1, \ldots, v_r)$ of dimension r. Then there exists an element $g \in K \setminus \{0\}$ such that $h = c \cdot f_{H\cap E}(\pi_H(\operatorname{Log} g))$ and $\prod_j \mathfrak{p}_j^{v_j} = \langle g \rangle$.*

Proof. We first observe that the result holds for the vectors of the basis B_L. For the r vectors on the right of B_L, this holds by construction. For the $r_1 + r_2 - 1$ vectors on the left, we have that $\prod_j \mathfrak{p}_j^0 = R = \langle u \rangle$ for any unit $u \in R$. So by definition of B_Λ, the property of Lemma 3.2 also holds for the $r_1 + r_2 - 1$ first vectors of B_L.

To complete the proof, it suffices to observe that the property of Lemma 3.2 is preserved by addition (if g_1 corresponds to a vector w_1 and g_2 corresponds to a vector w_2, then $g_1 g_2$ corresponds to the vector $v_1 + v_2$) and by multiplication by -1 (if g corresponds to a vector w, then g^{-1} corresponds to the vector $-w$). All these elements g are invertible as they are obtained by multiplying and inverting non-zero elements of K. □

3.3.2 Computation of the lattice L

The lattice L described above only depends on the number field we are working on. A basis of it can then be computed in a pre-processing phase, before the knowledge of the ideal in which we want to find a short non-zero vector. In this subsection, we give an algorithm to compute the lattice L and we show that this algorithm can be performed in time at most exponential in $\log |\Delta|$. As we shall see, this will even be sub-exponential in $\log |\Delta|$. The costly part of the pre-processing phase will be the pre-processing used for the CVPP algorithm.

Algorithm 3.1 Computes a basis B_L as described above

Input: A number field K and an integer $r = \operatorname{poly}(\log |\Delta|)$ such that $\log h_K \leq r$.
Output: The basis B_L described in Section 3.3.1.

1: Compute the set $\mathfrak{B}'$ of all prime ideals of algebraic norm $\leq 12 \log^2 |\Delta|$.
2: Compute all the relations between the elements of $\mathfrak{B}'$ and the log-unit lattice Λ.
3: Use the relations to extract a set $\mathfrak{B}'' \subseteq \mathfrak{B}'$ generating the class group with $|\mathfrak{B}''| \leq \log h_K$.
4: Compute the set $\mathfrak{P}$ of all prime ideals of norms smaller than some $\operatorname{poly}(\log |\Delta|)$ (choose the bound so that $|\mathfrak{P}| > r$).
5: Create a set $\mathfrak{B}$ by adding to $\mathfrak{B}''$ ideals taken uniformly in $\mathfrak{P}$, until the cardinality of $\mathfrak{B}$ reaches r.
6: Compute a basis of $\ker(f_{\mathfrak{B}})$ and generators g_i of the fractional principal ideals corresponding to the relations computed.
7: Create the matrix B_L from these r relations, the corresponding g_i and the log-unit lattice Λ computed at Step 2.
8: **return** B_L.

Lemma 3.3. *Assume GRH. Then Algorithm 3.1 outputs a matrix B_L as described above, in time at most*

$$T_{\text{log-unit}} + 2 \cdot T_{\text{rel}}(\operatorname{poly}(\log |\Delta|), \operatorname{poly}(\log |\Delta|)) + \operatorname{poly}(\log |\Delta|).$$

Proof. We analyze the cost of each step of the algorithm, and provide some details for the correctness when needed.

Step 1. We have already seen in Section 2.3.8 that computing all prime ideals of norm $\leq 12 \log^2 |\Delta|$ can be performed in time polynomial in $\log |\Delta|$. There are $\operatorname{poly}(\log |\Delta|)$ such ideals.

Step 2. Computing all the relations between the elements of $\mathfrak{B}'$ and the log-unit lattice Λ can be performed in time at most $T_{\text{rel}}(\text{poly}(\log|\Delta|), \text{poly}(\log|\Delta|)) + T_{\text{log-unit}}$. The relations between the elements of $\mathfrak{B}'$ are represented as an invertible matrix (whose columns span the kernel of the function $f_{\mathfrak{B}'}$ defined in Section 2.3.10).

Step 3. Extracting a generating set $\mathfrak{B}''$ from $\mathfrak{B}'$ of cardinality at most $\log h_K$ can be done using Lemma 2.12. Because we already have the matrix of relations between the elements of $\mathfrak{B}'$ (and because the size of this matrix is polynomial in $\log|\Delta|$), this can be done in polynomial time (as stated in the lemma).

Step 4. As in Step 1, this can be done in polynomial time, because the bound on the norms of the ideals is polynomial. We obtain a set $\mathfrak{P}$ whose cardinality is polynomial in $\log|\Delta|$.

Step 5. Picking uniform elements in a set of polynomial size can be done efficiently, so this step can be performed in polynomial time (recall that $r = \text{poly}(\log|\Delta|)$). Note that in the previous step, we had that the cardinality of $\mathfrak{B}'$ was at most $\log h_K \leq r$, so we can indeed add ideals to it to reach a set of cardinality r.

Step 6. As in Step 2, computing the kernel of $f_{\mathfrak{B}}$ can be done in time $T_{\text{rel}}(\text{poly}(\log|\Delta|), \text{poly}(\log|\Delta|)$. Together with the relations, we also get generators of the corresponding principal ideals.

Step 7. Finally, to compute the matrix B_L, we just need to compute the functions π_H and $f_{H\cap E}$ on the g_i's computed in Step 6. We then put it together with the matrix of relations computed in Step 6 and the log-unit lattice computed in Step 2. This can be done in polynomial time. □

3.3.3 From SVP in ideal lattices to CVP in L

We now explain how to transform the problem of finding a short non-zero vector in a fractional ideal I of R, into solving a CVP instance with respect to the lattice L described in Section 3.3.1. As explained above, the main idea is to multiply the ideal I by ideals of the set $\mathfrak{B}$, until we obtain a 'good' principal ideal (i.e., with a short generator). In terms of lattice operations in L, our initial lattice I will give us a target vector in the real vector space spanned by L. Multiplying it by ideals of $\mathfrak{B}$ will be the same as adding to the target a vector of the lattice L. Finally, checking whether the resulting ideal is a good principal ideal can be done by checking whether the obtained vector is short. Overall, we are indeed solving a CVP instance in L. We first describe the algorithm, and then prove its correctness and bound its cost.

Algorithm 3.2 Solves ideal SVP using an oracle to solve CVP in L

Input: A non-zero fractional ideal $I \subseteq R$ (given by some basis), the basis B_L defined above and some parameter $\beta = \beta(n) > 0$.

Output: A somehow short non-zero element in I.

1: Compute $v_1, \ldots, v_r \in \mathbb{Z}$ and $g \in K$ such that $I = \prod_j \mathfrak{p}_j^{v_j} \cdot \langle g \rangle$.
2: Let $t = (-c \cdot f_{H\cap E}(h_g), v_1 + \beta, \ldots, v_r + \beta)^T$, where $h_g = \pi_H(\text{Log}\, g)$.
3: Compute $w \in L$ such that $\|t - w\|_\infty \leq \beta$ (see Section 3.4).
4: Let $g' \in K$ be the element associated to w as in Lemma 3.2.
5: **return** $g \cdot g'$.

Theorem 3.4. *Let us fix $c = n^{1.5}/r$. Let $\beta = \beta(n) > 0$. Then, for any non-zero fractional ideal I of R, Algorithm 3.2 runs in time at most*

$$T_{\text{decomp}}(\mathcal{N}(I), \text{poly}(\log|\Delta|), \text{poly}(\log|\Delta|)) + T_{\text{CVP}}(\infty, L, \beta) + \text{poly}(\log|\Delta|)$$

and outputs a non-zero element $x \in I$ such that $\|x\|_2 \leq 2^{O(\frac{\beta \cdot r \cdot \log\log|\Delta|}{n})} \cdot \mathcal{N}(I)^{1/n}$.

Observe that in the statement of the run-time, the term $T_{\text{CVP}}(\infty, L, \beta)$ will be infinite if β is smaller than $\mu^{(\infty)}(L)$ (no algorithm can find a point of L at distance at most β given as input an arbitrary target vector). In this case, the run-time of our algorithm might also be infinite (i.e. the algorithm fails).

Proof. **Correctness.** Let us define the fractional ideal $J = \langle g \cdot g' \rangle$. This will be our 'good' principal ideal, i.e., a principal ideal with a small generator, and contained in I. Let us first prove that J is a multiple of I. By Lemma 3.2, we have $w = (c \cdot f_{H\cap E}(\pi_H(\operatorname{Log} g')), v'_1, \ldots, v'_r)^T$ with $\langle g' \rangle = \prod_j \mathfrak{p}_j^{v'_j}$. We can then write

$$\begin{aligned} J &= I \cdot \prod_j \mathfrak{p}_j^{-v_j} \cdot \langle g' \rangle && \text{by definition of } g \text{ and the } v_j\text{'s} \\ &= I \cdot \prod_j \mathfrak{p}_j^{-v_j} \cdot \prod_j \mathfrak{p}_j^{v'_j} && \text{by Lemma 3.2} \\ &= I \cdot \prod_j \mathfrak{p}_j^{v'_j - v_j}. \end{aligned}$$

Further, we know that $\|t - w\|_\infty \leq \beta$, and hence we have $v_j \leq v'_j \leq v_j + 2\beta$ for all j. In particular, we have that $v'_j - v_j \geq 0$ and so the ideal $\prod_j \mathfrak{p}_j^{v'_j - v_j}$ is an integral ideal. We conclude that J is contained in I, and in particular $g \cdot g'$ is indeed an element of I. Also, because $g' \neq 0$ (see Lemma 3.2) and $g \neq 0$ (we chose I to be non-zero), then $g \cdot g'$ is a non-zero element of I.

Let us now show that $g \cdot g'$ is short. We will do so by using Lemma 2.17. Let $\operatorname{Log} g = h_g + a_g \mathbf{1}$ and $\operatorname{Log} g' = h_{g'} + a_{g'} \mathbf{1}$ with h_g and $h_{g'} \in H \cap E$ (note that because $g, g' \in K$, we do not necessarily have $a_g, a_{g'} > 0$). We then have that $\operatorname{Log}(gg') = (h_g + h_{g'}) + (a_g + a_{g'})\mathbf{1}$. By Lemma 2.17, we know that $\|gg'\|_2 \leq \sqrt{n} \cdot |\mathcal{N}(gg')|^{1/n} \cdot 2^{\|h_g + h_{g'}\|_2}$. Therefore, it suffices to bound the two terms $|\mathcal{N}(gg')|^{1/n}$ and $\|h_g + h_{g'}\|_2$.

Let us start by $|\mathcal{N}(gg')|^{1/n}$. By multiplicativity of the algebraic norm, we have that $|\mathcal{N}(gg')|^{1/n} = \mathcal{N}(J)^{1/n} = \mathcal{N}(I)^{1/n} \cdot \prod_j \mathcal{N}(\mathfrak{p}_j)^{\frac{v'_j - v_j}{n}}$. We have chosen the ideals $\mathfrak{p}_j$ with polynomially bounded algebraic norms, and we have seen that $0 \leq v'_j - v_j \leq 2\beta$. Thus, we obtain that $\mathcal{N}(\mathfrak{p}_j)^{\frac{v'_j - v_j}{n}} = 2^{O(\frac{\beta \log\log|\Delta|}{n})}$. By taking the product of the r ideals $\mathfrak{p}_j$, we obtain

$$|\mathcal{N}(gg')|^{1/n} = \mathcal{N}(I)^{1/n} \cdot 2^{O(\frac{\beta \cdot r \cdot \log\log|\Delta|}{n})}.$$

We now consider the term $\|h_g + h_{g'}\|_2$. Recall that $\|w - t\|_\infty \leq \beta$, so in particular, if we consider only the first $r_1 + r_2 - 1$ coefficients of the vectors, we have that $\|c \cdot f_{H\cap E}(h_{g'}) + c \cdot f_{H\cap E}(h_g)\|_\infty \leq \beta$. And if we consider the ℓ_2-norm, we obtain $\|f_{H\cap E}(h_{g'}) + f_{H\cap E}(h_g)\|_2 \leq \sqrt{n}\beta/c$. Using the fact that the ℓ_2-norm is invariant by $f_{H\cap E}$, we conclude that $\|h_{g'} + h_g\|_2 \leq \sqrt{n}\beta/c$.

Finally, combining the two upper bounds above and replacing c by $n^{1.5}/r$, we obtain that

$$\|gg'\|_2 \leq \sqrt{n} \cdot 2^{O(\frac{\beta \cdot r \cdot \log\log|\Delta|}{n})} \cdot \mathcal{N}(I)^{1/n}.$$

Cost. Step 1 can be performed in time $T_{\text{decomp}}(\mathcal{N}(I), \text{poly}(\log|\Delta|), \text{poly}(\log|\Delta|))$. Step 2 can be performed in polynomial time. Step 3 uses a CVP solver and can be done in time $T_{\text{CVP}}(\infty, L, \beta)$. Finally, Step 4 only consists in recovering g' from the vector w, it can be done in polynomial time. Note that for this last step, if we only have the vector w, then we know $\pi_H(\operatorname{Log} g')$, but it might not be possible to recover g' from it. On the other hand, the lower part of the vector w also gives us the ideal $\langle g' \rangle$, but then computing g' from it would be costly. In order to perform this step in polynomial time, when creating the matrix B_L we keep in memory the elements g_i corresponding to the different columns. Then, when we obtain w, we only have to write it as a linear combination of the vectors of B_L and we can recover g' as a product of the g_i's. This can also be done in polynomial time.

□

Combining Algorithm 3.2 with the pre-processing phase (i.e., computing B_L with Algorithm 3.1 and pre-processing it for approx-CVPP'), we obtain the following theorem.

Theorem 3.5. *Let K be any number field of dimension n and discriminant Δ. Let $\alpha \in [0, 1]$, $r = \text{poly}(\log|\Delta|)$ be such that $\log h_K \leq r$, and $\nu := r + r_1 + r_2 - 1$. Then, under GRH, there exist two algorithms $A_{\text{pre-proc}}$ and A_{query} such that*

- *Algorithm $A_{\text{pre-proc}}$ takes as inputs the field K and a basis of its ring of integers R, runs in time*

$$T_{\text{CVP}}^{\text{pre-proc}}(\infty, L, \nu^\alpha) + T_{\text{log-unit}} + 2 \cdot T_{\text{rel}}(\text{poly}(\log|\Delta|), \text{poly}(\log|\Delta|)) + \text{poly}(\log|\Delta|)$$

and outputs a hint w of bit-size at most $T_{\text{CVP}}^{\text{query}}(\infty, L, \nu^\alpha)$;

- *Algorithm A_{query} takes as inputs the hint w output by $A_{\text{pre-proc}}$ and any fractional ideal I of R such that the numerator and denominator of $\mathcal{N}(I)$ have bit-sizes bounded by $\text{poly}(\log|\Delta|)$; it runs in time*

$$T_{\text{decomp}}(\mathcal{N}(I), \text{poly}(\log|\Delta|), \text{poly}(\log|\Delta|)) + T_{\text{CVP}}^{\text{query}}(\infty, L, \nu^\alpha) + \text{poly}(\log|\Delta|)$$

and outputs a non-zero element $x \in I$ such that

$$\|x\|_2 \leq 2^{O(\frac{\nu^\alpha \cdot r \cdot \log\log|\Delta|}{n})} \cdot \lambda_1^{(2)}(I).$$

The lattice L is as defined in Section 3.3.1 and only depends on the field K. The memory consumption of both algorithms is bounded by their run-times.

Note that we used the fact that $\lambda_1^{(2)}(I) \geq \lambda_1^{(\infty)}(I) \geq \mathcal{N}(I)^{1/n}$ (see Inequality (2.12)) to replace the $\mathcal{N}(I)^{1/n}$ term in Theorem 3.4 by $\lambda_1^{(2)}(I)$.

3.4 Solving CVP' with Pre-processing

In this section, we describe a possible way of solving approx-CVP' in the lattice L defined previously. Even if our lattice L has some structure, it does not seem easy to solve approx-CVP' in it (not necessarily easier than solving the approx-SVP instance directly for the initial lattice I). However, the lattice L only depends on the field K and not on the ideal I. Hence, in this section, we focus on solving approx-CVP' with pre-processing on the lattice (to which we refer as CVPP'). Combining it with the result of Section 3.3, this will provide an algorithm to solve approx-SVP in ideals, with pre-processing on the field K.

3.4.1 Properties of the lattice L

Recall that our lattice L is given by the basis matrix $B_L = \left(\begin{array}{c|c} c \cdot B_\Lambda & A_{\mathfrak{B}} \\ \hline 0 & R_{\mathfrak{B}} \end{array}\right) \in \mathbb{R}^{\nu \times \nu}$, where we let $A_{\mathfrak{B}}$ denote the top-right block of B_L consisting of the vectors $c \cdot \widetilde{h}_{g_i}$, and $R_{\mathfrak{B}}$ be the bottom-right block of B_L containing the relations of the elements of $\mathfrak{B}$. Recall that $R_{\mathfrak{B}}$ is a basis of the kernel of $f_{\mathfrak{B}} : (e_1, \ldots, e_r) \in \mathbb{Z}^r \mapsto [\prod_j \mathfrak{p}_j^{e_j}] \in Cl_K$. Hence we have $\det(R_{\mathfrak{B}}) = |\mathbb{Z}^r / \ker(f_{\mathfrak{B}})| = h_K$.

Equation (2.14) gives us that $\det(\Lambda) \cdot h_K \leq 2^{O(\log|\Delta| + n\log\log|\Delta|)}$. Hence, we have that $\det(L) = c^{r_1+r_2-1} \cdot 2^{O(\log|\Delta| + n\log\log|\Delta|)}$. We chose $c = n^{1.5}/r$ in Theorem 3.4. We then obtain the following upper bound on $\det(L)$:

$$\det(L) = \left(\frac{n^{1.5}}{r}\right)^{r_1+r_2-1} \cdot 2^{O(\log|\Delta| + n\log\log|\Delta|)} = 2^{O(\log|\Delta| + n\log\log|\Delta|)}.$$

We still have some freedom for the choice of the parameter r (and hence the dimension $\nu = r + r_1 + r_2 - 1$ of the lattice L), as long as $\log h_K \leq r$. We will choose it sufficiently large to ensure that the root determinant of L is at most constant. On the other hand, the dimension of L should be as small as possible as it impacts the cost of the CVP computations. We fix

$$r = \max(\log h_K, \log|\Delta| + n\log\log|\Delta|).$$

This choice of r satisfies $r \geq \log h_K$ and $\det(L)^{1/\nu} \leq O(1)$. Note that as $\log h_K = \widetilde{O}(\log|\Delta|)$ (see Equation (2.13)), we have $r \leq \widetilde{O}(\log|\Delta|)$.

In the following, we view the lattice L as random, where the randomness comes from the choice of the set $\mathfrak{B}$ (the initial set $\mathfrak{B}''$ in Algorithm 3.1 is fixed, but then we add to it random prime ideals of

polynomially bounded norms to create the set $\mathfrak{B}$). If the created lattice L does not satisfy the conditions we want, we can try another lattice by sampling a new set $\mathfrak{B}$. As we chose r so that $\det(L)^{1/\nu} = O(1)$, we know by Minkowski's inequality that $\lambda_1^{(\infty)}(L) = O(1)$. Then, because L is somehow random, we also expect that all successive minima $\lambda_i^{(\infty)}(L)$ and the covering radius in infinity norm are constant. Hence, we expect to be able to take β as small as $O(1)$ in Algorithm 3.2. We summarize this assumption below.

Heuristic 3.6. *With good probability over the choice of $\mathfrak{B}$, the ℓ_∞-norm covering radius of L satisfies $\mu^{(\infty)}(L) = O(1)$ (and hence $\mu^{(2)}(L) = O(\sqrt{\nu})$).*

This heuristic assumption calls for a few comments. First, it is better analyzed as two separate conjectures, one on the log-unit lattice, the other one on the class group lattice. Concerning the latter, assume that the class number is a prime p. Then we can choose $\mathfrak{p}_1$ to be a generator of the class group, and the relation matrix is of the form

$$\begin{pmatrix} p & a_1 & a_2 & \dots & a_r \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix},$$

where the a_i's in $[0, p-1]$ characterize the elements of the ideal class group. In our setting where each $\mathfrak{p}_i$ (for $i \geq 2$) is picked randomly among small prime ideals, we can thus reasonably assume that the a_i's are uniformly distributed in $[0, p-1]$. Hence, for $(e_i)_{2 \leq i \leq r} \in [-B, B]^{r-1}$ for some constant $B \geq 1$, we can expect that one among the $(2B+1)^{r-1} = p^c$ (with $c > 1$) fractional ideals $\prod_{i \geq 2} \mathfrak{p}_i^{e_i}$ is in a class $[\mathfrak{p}_1]^a$ for some $a = O(1)$, which implies that the ℓ_∞-norm covering radius is $O(1)$.

The general case is analogous to this first intuition. Let $\mathfrak{B} = \{\mathfrak{p}_1, \dots, \mathfrak{p}_s, \mathfrak{p}_{s+1}, \dots, \mathfrak{p}_r\}$ with $\{\mathfrak{p}_1, \dots, \mathfrak{p}_s\}$ the prime ideals coming from the set $\mathfrak{B}''$ (hence fixed) and $\{\mathfrak{p}_{s+1}, \dots, \mathfrak{p}_r\}$ the ideals uniformly chosen among prime ideals of norm bounded by some polynomial. Because the set $\mathfrak{B}''$ generates the class group, we can find a basis of L of the following form, by taking the HNF matrix for the bottom-right part of B_L.

$$B'_L := \begin{array}{c|cc} \overbrace{\quad}^{r_1+r_2-1} & \overbrace{\qquad\qquad}^{r} & \\ \hline c \cdot B_\Lambda & c \cdot \widetilde{h}_{g_1}, \dots, c \cdot \widetilde{h}_{g_r} & \updownarrow\, r_1+r_2-1 \\ \hline & R_{\mathfrak{B}''} \quad v_{s+1} \cdots v_r & \updownarrow\, s \\ 0 & 1 & \\ & \ddots & \updownarrow\, r \\ & 1 & \end{array}$$

In this matrix, the block matrices B_Λ and $R_{\mathfrak{B}''}$ are fixed, as well as the vectors $\widetilde{h}_{g_i}$ for i in $\{1, \dots, s\}$. However, the vectors v_i and $\widetilde{h}_{g_i}$ for $s < i \leq r$ depend on our choices of $\{\mathfrak{p}_{s+1}, \dots, \mathfrak{p}_r\}$. The vectors of $\mathbb{Z}^s / R_{\mathfrak{B}''}$ are in bijection with the elements of the ideal class group (because $\mathfrak{B}''$ generates the class group). So if we assume that the class of a uniform prime ideal of norm polynomially bounded is uniform in the class group, then we would have that the vectors v_i of the matrix above are uniform in $\mathbb{Z}^s / R_{\mathfrak{B}''}$. In a similar way, we will assume that the vectors $\widetilde{h}_{g_i}$ are somehow uniform in $\mathbb{R}^{r_1+r_2-1}/\Lambda$ (recall that they correspond to the projection over H of $\operatorname{Log} g_i$ for g_i a generator of the principal ideal associated with the lower part of the vector).

Let us now explain why, given any target vector $t \in \mathbb{R}^\nu$, we expect to find a vector $v \in L$ at distance $O(1)$ from t. Write $t = (c \cdot \widetilde{h}, v, w)^T$ with $\widetilde{h}$ of dimension $r_1 + r_2 - 1$, v of dimension s and w of

Conductor of K	Dimension of L	$\widetilde{\mu}^{(2)}(L)$	$\widetilde{\mu}^{(\infty)}(L)$
18	9	1.13	0.755
16	16	1.50	0.899
36	28	1.79	0.795
40	41	2.15	0.893
48	42	2.19	0.840
32	44	2.26	0.794
27	49	2.36	0.901
66	54	2.47	0.989
44	57	2.53	0.815
70	67	2.72	1.03
84	68	2.74	1.27
90	68	2.71	0.814
78	70	2.81	0.882
72	73	2.90	1.00

Table 3.1: Approximate covering radii in ℓ_2 and ℓ_∞ norms for the lattice L, for cyclotomic number fields of different conductors.

dimension $r-s$. We can assume, without loss of generality, that $|w_i| < 1/2$ for all i (using the last $r-s$ columns of B'_L to reduce t if needed). By taking the subset sums of the last $r-s$ columns of B'_L, we obtain 2^{r-s} vectors of L of the form $t' = (c \cdot \widetilde{h}', v', w')^T$, with $w' \in \{0,1\}^{r-s}$. Because we assumed that the v_i and $\widetilde{h}_{g_i}$ for $s < i \leq r$ were somehow uniform modulo $R_{\mathfrak{B}''}$ and Λ respectively, we also expect the vectors $\widetilde{h}'$ and v' created above to be somehow uniform modulo $R_{\mathfrak{B}''}$ and Λ. Recall that we chose r so that $(\det(c\Lambda) \cdot \det(R_{\mathfrak{B}''}))^{1/r} = O(1)$, hence the volume of $c\Lambda$ and $R_{\mathfrak{B}''}$ satisfies $\det(c\Lambda) \cdot \det(R_{\mathfrak{B}''}) \leq 2^{O(r)}$. We can then assume that we have $2^{r-s} > \det(c\Lambda) \cdot \det(R\mathfrak{B}'')$ (if needed, we can multiply r by a constant factor, which will not change the asymptotics). This means that we expect to find one of the 2^{r-s} vector $t' = (c \cdot \widetilde{h}', v', w')^T$ satisfying $\|(c \cdot \widetilde{h}, v) - (c \cdot \widetilde{h}', v')\|_\infty = O(1)$. And because $|w_i| < 1/2$ and $w'_i \in \{0,1\}$ we also have $\|t - t'\|_\infty = O(1)$.

We experimentally computed the lattice L for some cyclotomic fields (using Algorithm 3.1). For each lattice L, we then computed an empirical covering radius. To do so, we picked 21 random target vectors t_i in the real span of the lattice. These vectors were sampled from a continuous Gaussian distribution with standard deviation $\sigma = 100$.

We then solved the CVP instances in L for these target vectors t_i and let v_i be a closest vector in L (for the ℓ_2-norm). We defined $\widetilde{\mu}^{(2)}(L)$ to be $\max_i \|t_i - v_i\|_2$ and $\widetilde{\mu}^{(\infty)}(L)$ to be $\max_i \|t_i - v_i\|_\infty$.[4] The approximated values of $\mu^{(2)}(L)$ and $\mu^{(\infty)}(L)$ are given in Table 3.1.

We observe that, while $\widetilde{\mu}^{(2)}(L)$ increases with the dimension (we expect that it increases as $\sqrt{\nu}$), the approximate covering radius in ℓ_∞-norm $\widetilde{\mu}^{(\infty)}(L)$ seems to remain constant around 1. These experimental results are consistent with Heuristic 3.6.

3.4.2 Using Laarhoven's algorithm

We now consider Laarhoven's algorithm, which solves approx-CVPP in Euclidean norm. While we would prefer an algorithm which solves approx-CVPP' in infinity norm, we only found algorithms for the Euclidean norm in the literature.

Recall from Section 2.2.1 that, for a ν-dimensional lattice L, Laarhoven's (heuristic) algorithm gives, for any $\alpha \in [0, 1/2]$:

$$\begin{aligned} T_{\text{CVP}}^{\text{pre-proc}}(2, L, O(\nu^\alpha) \cdot \mu^{(2)}(L)) &= 2^{O(\nu)}, \\ T_{\text{CVP}}^{\text{query}}(2, L, O(\nu^\alpha) \cdot \mu^{(2)}(L)) &= 2^{\widetilde{O}(\nu^{1-2\alpha})}. \end{aligned}$$

[4] As we solved CVP in L for the ℓ_2-norm, the quantity $\mu^{(\infty)}(L)$ may be over-estimated, but this should not be over-estimated by too much. Further, as we want an upper bound on $\mu^{(\infty)}(L)$, this is not an issue.

As we have assumed (Heuristic 3.6) that $\mu^{(2)}(L) = O(\sqrt{\nu})$ with good probability over the choice of L, this implies that Laarhoven's algorithm achieves

$$T_{\mathrm{CVP}}^{\text{pre-proc}}(2, L, O(\nu^{1/2+\alpha})) = 2^{O(\nu)} \quad \text{and} \quad T_{\mathrm{CVP}}^{\text{query}}(2, L, O(\nu^{1/2+\alpha})) = 2^{\widetilde{O}(\nu^{1-2\alpha})}.$$

We now have an algorithm that, given any input $t \in \mathrm{Span}(L)$, outputs a vector $v \in L$ such that $\|t-v\|_2 \leq O(\nu^{1/2+\alpha})$, while we would like to have $\|t-v\|_\infty \leq O(\nu^\alpha)$. However, when we pick a random vector of Euclidean norm bounded by $O(\nu^{1/2+\alpha})$, we expect that with good probability, its coefficients are somehow balanced. Hence we expect its infinity norm to be approximately $\sqrt{\nu}$ times smaller than its Euclidean norm. This is the meaning of the following heuristic assumptions.

First, because we want the output of Laarhoven's algorithm to be somehow random, we argue that we can randomize the input vector of the CVPP algorithm.

Heuristic 3.7. *We assume that in our algorithm, the target vector t given as input to Laarhoven's algorithm behaves as a random vector sampled uniformly in* $\mathrm{Span}(L)/L$.

This assumption that t is distributed uniformly in $\mathrm{Span}(L)/L$ may be justified by the fact that, in Algorithm 3.2, we can somehow randomize our target vector t by multiplying our initial fractional ideal I by an integral ideal of small algebraic norm (statistically independent of the $\mathfrak{p}_i$'s chosen for $\mathfrak{B}$).

Heuristic 3.8. *With non-negligible probability over the input target vector t, distributed uniformly in* $\mathrm{Span}(L)/L$*, the vector v output by Laarhoven's algorithm satisfies $\|t-v\|_\infty \leq \widetilde{O}(\|t-v\|_2/\sqrt{\nu})$.*

In order to motivate Heuristic 3.8, we recall that if a vector is drawn uniformly at random on a sphere, then its ℓ_∞-norm is smaller than its ℓ_2-norm by a factor $O(\log n/\sqrt{n})$, with good probability.

Lemma 3.9. *Let x be sampled uniformly on the unit sphere S^{n-1} in $\mathbb{R}^n$. Then* $\Pr(\|x\|_\infty \geq \frac{\sqrt{8}\ln n}{\sqrt{n}}) \leq O(\frac{1}{\sqrt{\ln n}})$.

Proof. Sampling x uniformly in S^{n-1} is the same as sampling y from a centered spherical (continuous) Gaussian distribution of parameter 1 and then normalizing it by setting $x = \frac{y}{\|y\|_2}$. So we have $\|x\|_\infty = \frac{\|y\|_\infty}{\|y\|_2}$, and it is sufficient to find an upper bound on $\|y\|_\infty$ and a lower bound on $\|y\|_2$. We know that for a centered spherical Gaussian distribution of parameter 1, we have $\Pr(\|y\|_\infty > 2\ln n) = \Pr(\exists i : |y_i| > 2\ln n) \leq \frac{1}{2\sqrt{2\pi \ln n}}$. Moreover, we also have that $\Pr(\|y\|_2 < \sqrt{n/2}) \leq \mathrm{e}^{-n/8}$ (see for instance [LM00, Lemma 1]). By the union bound, we finally obtain that $\Pr(\|y\|_\infty/\|y\|_2 > \frac{\sqrt{8}\ln n}{\sqrt{n}}) \leq O(\frac{1}{\sqrt{\ln n}})$. □

Note that the proof also shows that for a vector y following a continuous Gaussian distribution of dimension n, we have $\|y\|_\infty/\|y\|_2 = O(\log n/\sqrt{n})$ with good probability. We also have experimental results corroborating Heuristic 3.8. We implemented our algorithm in Magma, both the generation of the lattice L and the CVPP phase using Laarhoven's algorithm. We tested our implementation for different cyclotomic fields. The maximum conductor achieved was 90. The maximum dimension of the lattice L that we achieved was 73, for a cyclotomic field of conductor 72. For these cyclotomic fields, we computed the lattice L. Then, we sampled target vectors t in the real span of L, using a Gaussian distribution of parameter $\sigma = 100$, and we ran Laarhoven's CVP algorithm to obtain a vector $v \in L$. We then computed the ratios $\frac{\|t-v\|_2}{\|t-v\|_\infty}$, which we expect to be around $O(\sqrt{\nu}/\log\nu)$. Because we are working in small dimensions, the $\log\nu$ term has a non-negligible impact. So, instead of plotting $\log(\frac{\|t-v\|_2}{\|t-v\|_\infty})$ as a function of $\log\nu$, we compared our ratios with the ones we would have obtained if the vectors were Gaussian vectors. On Figure 3.3, the blue dots represent the logarithms of the ratios $\frac{\|t-v\|_2}{\|t-v\|_\infty}$ obtained when choosing a random Gaussian vector t as input of our algorithm. For every fixed conductor, we have several vertically aligned points, because we tried Laarhoven's algorithm for different approximation factors (i.e., different choices of α). The green '+' are obtained by computing $\log(\|x\|_2/\|x\|_\infty)$ for some Gaussian vectors of dimension ν. The red crosses are obtained by taking the median point of a large number of green '+' (not all of them are plotted on the figure). We observe that the ratios obtained with our algorithm are well aligned with the red crosses. Moreover, even if we have some variance within the blue dots, it is comparable to the variance observed within the green '+'. So Heuristic 3.8 seems consistent with our empirical experiments (recall that Gaussian vectors provably satisfy Heuristic 3.8 with good probability).

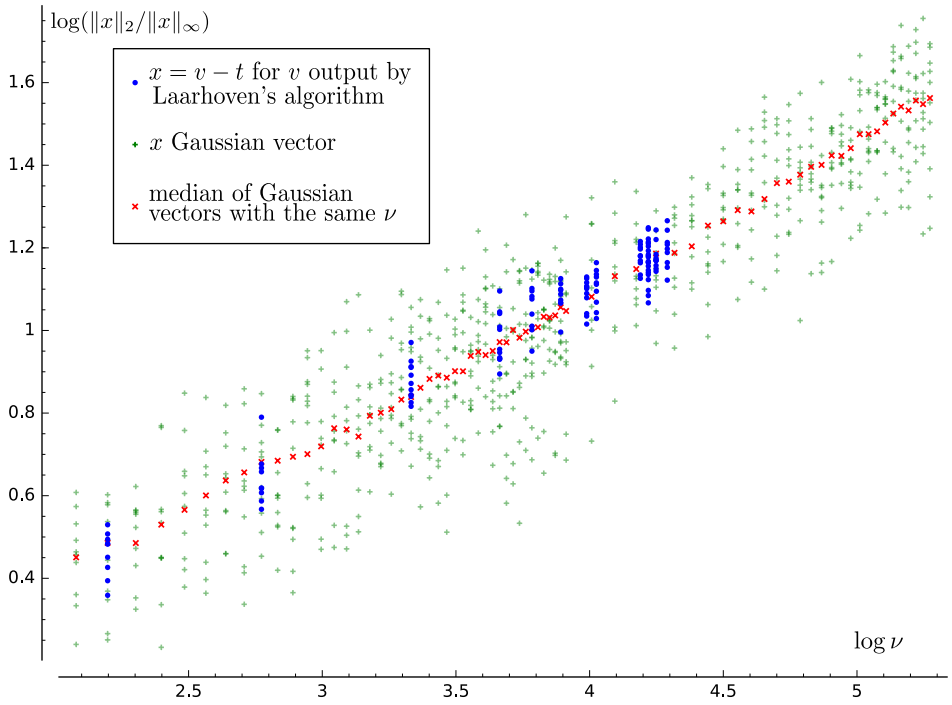

Figure 3.3: Comparison of $\log(\|x\|_2/\|x\|_\infty)$ as a function of $\log \nu$ for x a Gaussian vector or $x = t - v$ with t a random target and v the approx-CVP solution output by Laarhoven's algorithm (on our lattice L, in selected cyclotomic fields).

We conclude that, under Heuristics 3.6, 3.7 and 3.8, and Heuristic 2.7 present in [Laa16], for any $\alpha \in [0, 1/2]$, Laarhoven's algorithm solves approx-CVPP' with

$$T_{\text{CVP}}^{\text{pre-proc}}(\infty, L, \nu^\alpha) = 2^{O(\nu)} \quad \text{and} \quad T_{\text{CVP}}^{\text{query}}(\infty, L, \nu^\alpha) = 2^{\widetilde{O}(\nu^{1-2\alpha})}. \tag{3.1}$$

3.5 Instantiating Theorem 3.5

We now instantiate Theorem 3.5 with $\nu = \widetilde{O}(\log \Delta)$ and the values given in Section 2.3.10 and in Equation (3.1) for $T_{\text{log-unit}}, T_{\text{decomp}}, T_{\text{rel}}, T_{\text{CVP}}^{\text{pre-proc}}$ and $T_{\text{CVP}}^{\text{query}}$.

Theorem 3.10. *Let K be any number field of dimension n and discriminant Δ. Let $\alpha \in [0, 1/2]$. Then, under GRH and Heuristics 2.7, 2.18, 2.19, 3.6, 3.7 and 3.8, there exist two algorithms $A_{\text{pre-proc}}$ and A_{query} such that*

- *Algorithm $A_{\text{pre-proc}}$ takes as inputs the field K and a basis of its integer ring R, runs in time $2^{\widetilde{O}(\log|\Delta|)}$ and outputs a hint w of bit-size at most $2^{\widetilde{O}((\log|\Delta|)^{1-2\alpha})}$,*
- *Algorithm A_{query} takes as inputs the hint w output by $A_{\text{pre-proc}}$ and any fractional ideal I of R such that the numerator and denominator of $\mathcal{N}(I)$ have bit-sizes bounded by $\text{poly}(\log|\Delta|)$. It runs in classical time $2^{\widetilde{O}((\log|\Delta|)^{\max(2/3,1-2\alpha)})}$ or in quantum time $2^{\widetilde{O}((\log|\Delta|)^{1-2\alpha})}$ and outputs an element $x \in I$ such that $0 < \|x\|_2 \leq 2^{\widetilde{O}(\frac{(\log|\Delta|)^{\alpha+1}}{n})} \cdot \lambda_1^{(2)}(I)$.*

The memory consumption of both algorithms is bounded by their run-times.

In the case where $\log|\Delta| = \widetilde{O}(n)$, we can replace $\log|\Delta|$ by n in all the equations of Theorem 3.10, and we obtain an element x which is a $2^{\widetilde{O}(n^\alpha)}$ approximation of a shortest non-zero vector of I (see Figure 3.4). On the other hand, if $\log|\Delta|$ becomes significantly larger than n, then both the run-time

and the approximation factor degrade. The cost of the pre-computation phase also becomes larger than $2^{O(n)}$. However, the query phase still improves upon the BKZ algorithm, for some choices of α, as long as $\log|\Delta| = \widetilde{O}(n^{12/11})$ in the classical setting or $\log|\Delta| = \widetilde{O}(n^{4/3})$ in the quantum setting (see Figure 3.5). In Figures 3.4 and 3.5, we plot the ratios between time and approximation factor for the query phase of our algorithm, in the different regimes $\log|\Delta| = \widetilde{O}(n)$ and $\log|\Delta| = \widetilde{O}(n^{1+\varepsilon})$ for some $\varepsilon > 0$.

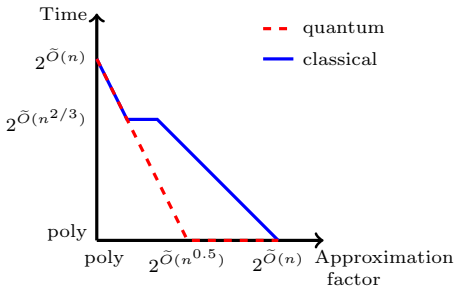

Figure 3.4: New trade-offs for ideal lattices in number fields satisfying $\log|\Delta| = \widetilde{O}(n)$ (with a pre-processing of cost $\exp(\widetilde{O}(n))$).

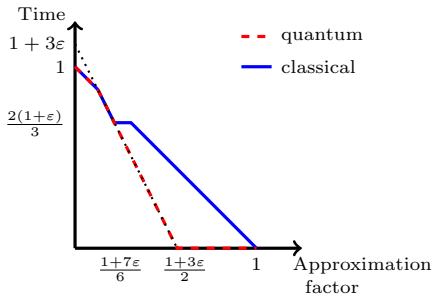

Figure 3.5: New trade-offs for ideal lattices in number fields satisfying $\log|\Delta| = \widetilde{O}(n^{1+\varepsilon})$ for some $\varepsilon > 0$ (with a pre-processing of cost $\exp(\widetilde{O}(n^{1+\varepsilon}))$).

In the case of prime-power cyclotomic fields, we know that $\log|\Delta| = \widetilde{O}(n)$. Moreover, there is a heuristic algorithm of Biasse *et al.* [BEF+17] satisfying $T_{\text{rel}}, T_{\text{decomp}}, T_{\text{log-unit}} \approx 2^{\widetilde{O}(n^{1/2})}$. Hence, we obtain the trade-offs shown in Figure 3.2 (in the introduction) when applying our algorithm to prime-power cyclotomic fields. Recall that in this special case, we already had an improvement upon the BKZ algorithm in the quantum setting, using the results of [CDPR16] and [CDW17], see Figure 3.1.

3.5.1 Using a CVP oracle in a fixed lattice

In the next chapter, we will be interested in algorithms which are allowed to query an oracle solving (exact) CVP in a fixed lattice depending only on the number field K. This means that we could replace Laarhoven CVPP algorithm in the theorem above by a call to an oracle solving CVP in the lattice L. We also extend slightly the statement of the theorem by considering lattices of the form αI for some fractional ideal I and element $\alpha \in K_{\mathbb{R}}$ (note that if α was in K, this would be a fractional ideal). Overall, we obtain the following lemma.

Lemma 3.11. *For any number field K, there exists a lattice L_K (that only depends on K and has dimension $\widetilde{O}(\log \Delta_K)$) such that, given an oracle access to an algorithm that solves* CVP *for L_K, the following holds. Under GRH and Heuristics 3.6, 3.7 and 3.12, there exists a quantum polynomial-time algorithm that takes as input an ideal I of K and any $\alpha \in K_{\mathbb{R}}^{\times}$, and outputs $x \in \alpha I \setminus \{0\}$ such that*

$$\|x\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(\alpha)|^{1/n} \cdot \mathcal{N}(I)^{1/n},$$

where $\mathrm{c} = 2^{\frac{\widetilde{O}(\log|\Delta|)}{n}}$. In particular, we have $\|x\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(x)|^{1/n}$.

Recall that the quantum computations can be replaced by classical ones (e.g., [BF14,Gel17,BEF+17]), at the expense of increased run-times and additional heuristic assumptions.

Because we are using an oracle to solve the closest vector problem in L (in Euclidean norm) instead of Laarhoven's algorithm, we need to modify Heuristic 3.8 in the following way.

Heuristic 3.12. *With non-negligible probability over the input target vector t, distributed uniformly in* $\mathrm{Span}(L)/L$*, the vector v which is the closest to t in L satisfies $\|t - v\|_\infty \leq \widetilde{O}(\|t - v\|_2/\sqrt{\nu})$.*

Proof. The lemma can be derived from Theorem 3.5 by taking $L_K = L$, $r = \widetilde{O}(\log|\Delta|)$ as in Section 3.4.1, $\alpha = 0$ and $T_{\text{CVP}}^{\text{pre-proc}}(\infty, L, \nu^\alpha) = T_{\text{CVP}}^{\text{pre-proc}}(\infty, L, \nu^\alpha) = \text{poly}(n)$ (because we assumed that we have an oracle solving CVP in $L_K = L$).

In Algorithm 3.2, the target vector t of the CVP computation is derived from the decomposition of $[I]$ on the $[\mathfrak{p}_i]$'s and the logarithm $\mathrm{Log}(g)$ of an element $g \in K$. To obtain the statement above, we replace $\mathrm{Log}(g)$ by $\mathrm{Log}(g \cdot \alpha) = \mathrm{Log}(g) + \mathrm{Log}(\alpha)$. The last lemma statement $\|x\|_\infty \leq \mathrm{c}|\mathcal{N}(x)|^{1/n}$ comes from the observation that $|\mathcal{N}(x)| \geq \mathcal{N}(\alpha) \cdot \mathcal{N}(I)$ (which holds because x belongs to $\alpha I \setminus \{0\}$). □

3.6 Conclusion

We have described in this chapter an algorithm which, after some exponential pre-processing, enables us to find somehow short vectors in ideal lattices more efficiently than the BKZ algorithm, for a large range of approximation factors. One may then wonder whether this algorithm can be improved, either in the pre-processing phase (can we obtain a pre-processing phase which is less than exponential?), or in the query phase (can we further improve the trade-offs?).

In order to better understand how the algorithm could be improved, we kept track of all the different sub-algorithms that compose our approx-SVP solver. The exact formulation of the total cost of the algorithm, as a function of the costs of the sub-algorithms, is given in Theorem 3.5. This formulation allows us to see whether an improvement of the run-time of one of them leads to an improvement of the overall cost of the approx-SVP solver. In particular, we observe that:

- Improving the approx-CVP solver would lead to an improvement of the slope of the curves in Figure 3.2, for approximation factors smaller than $2^{\widetilde{O}(\sqrt{n})}$. In a different direction, removing the pre-processing step needed for this approx-CVP solver would remove the pre-processing of the overall approx-SVP algorithm.
- Designing a classical algorithm that performs class group related computations in time less than $2^{\widetilde{O}(\sqrt{n})}$ would allow to further extend the (classical) segment of Figure 3.2 with slope $-1/2$, until it reaches the cost needed to solve these class group related problems. For example, Biasse described in [Bia17] an algorithm to solve the principal ideal problem in cyclotomic fields of prime-power conductor, with pre-processing. After pre-computations depending on the field only, this algorithm finds a generator of a principal ideal in time less than $2^{\widetilde{O}(\sqrt{n})}$ if the ideal has algebraic norm $\leq 2^{\widetilde{O}(n^{1.5})}$.

Finally, one could wonder whether it is possible to find significantly faster approx-SVP algorithms for specific families of number fields and/or restricted families of ideals. For instance, the Bauch *et al.* algorithm from [BBV+17] and the follow-up algorithm of Biasse and van Vredendaal [BV18] efficiently solve class group related problems in real multiquadratic number fields in the classical setting. This means that in these number fields, the classical version of our algorithm is as efficient as the quantum one (there is no threshold for the query phase in the classical setting). However, the algorithm still requires an exponential pre-processing phase for the approx-CVP solver.

Chapter 4

An LLL algorithm for module lattices

As mentioned in the previous chapter, it is not clear whether being able to find short vectors in ideal lattices has any impact on the difficulty of problems such as RingLWE or RingSIS, which are the problems whose intractability is most often used to support the security of most efficient lattice-based cryptographic schemes. However, it is known that these problems are equivalent to SIVP in module lattices [LS15, AD17]. A module of rank one is (essentially) an ideal, hence SIVP in module lattices includes SVP in ideal lattices.

In this chapter, we first focus on the simplest modules which are not ideals, i.e., modules of rank 2. We propose an algorithm which mimics the LLL algorithm (or Gauss's algorithm, as we are considering 2-dimensional matrices) for rank 2 modules over some ring of integers R instead of $\mathbb{Z}$. If given access to an oracle solving CVP in a fixed lattice depending only on the number field, our algorithm outputs a small element of the module in quantum polynomial time. In the case of a power-of-two cyclotomic number field of degree n, the fixed lattice in which we want to solve CVP has dimension roughly n^2 and the small element output by the algorithm is a quasi-polynomial approximation of the shortest vector of the module. This algorithm uses in a black box way the one described in the previous chapter. We will only use Lemma 3.11 from Chapter 3, hence this chapter can be read independently from the previous one. In a second time, we describe a generalization of the LLL algorithm to modules of arbitrary rank m, which uses as a black box the approx-SVP solver for rank-2 modules described above (or any other approx-SVP solver for rank-2 modules).

This chapter is based on a joint work with Changmin Lee, Damien Stehlé and Alexandre Wallet, which was accepted at Asiacrypt 2019 [LPSW19]. The code that was used to perform the experiments described in this chapter is available at

`http://perso.ens-lyon.fr/alice.pellet___mary/code/code-module-lll.zip`

Contents

4.1 Introduction

The NTRU [HPS98], RingSIS [LM06, PR06], RingLWE [SSTX09, LPR10], ModuleSIS and ModuleLWE [BGV14, LS15] problems and their variants serve as security foundations of numerous cryptographic protocols. Their main advantages are their presumed quantum hardness, their flexibility for realizing advanced cryptographic functionalities, and their efficiency compared to their SIS and LWE counterparts [Ajt96, Reg05]. As an illustration of their popularity for cryptographic design, we note that 11 out of the 26 candidates at Round 2 of the NIST standardization process for post-quantum cryptography rely on these problems or variants thereof.[1] From a hardness perspective, these problems are best viewed as standard problems on Euclidean lattices, restricted to random lattices corresponding to modules over the rings of integers of number fields. Further, for some parametrizations, there exist reductions from and to standard worst-case problems for such module lattices [LS15, AD17, RSW18].

Let K be a number field and R its ring of integers. In this introduction, we will use the power-of-2 cyclotomic fields $K = \mathbb{Q}[x]/(x^n+1)$ and their rings of integers $R = \mathbb{Z}[x]/(x^n+1)$ as a running example (with n a power of 2). An R-module $M \subset K^m$ is a finitely generated subset of vectors in K^m that is stable under addition and multiplication by elements of R. As an example, if we consider $h \in R/qR$ for some integer q, the set $\{(f,g)^T \in R^2 : fh = g \bmod q\}$ is a module. If h is an NTRU public key, the corresponding secret key is a vector in that module, and its entries have coefficients that are small. Note that for $K = \mathbb{Q}$ and $R = \mathbb{Z}$, we recover Euclidean lattices in $\mathbb{Q}^m$. A first difficulty for handling modules compared to lattices is that R may not be a Euclidean domain, and, as a result, a module M may not be of the form $M = \sum_i R\mathbf{b}_i$ for some linearly independent $\mathbf{b}_i$'s in M. However, as R is a Dedekind domain, for every module M, there exist K-linearly independent $\mathbf{b}_i$'s and fractional ideals I_i such that $M = \sum I_i\mathbf{b}_i$ (see, e.g., [O'M63, Th. 81:3]). The set $((I_i, \mathbf{b}_i))_i$ is called a pseudo-basis of M. A module in K^m can always be viewed as a lattice in $\mathbb{C}^{mn}$ by mapping elements of K to $\mathbb{C}^n$ via the canonical embedding map (for our running example, it is equivalent to mapping a polynomial of degree $< n$ to the vector of its coefficients). Reminders about modules can be found in Section 2.3.7.

Standard lattice problems, such as finding a full-rank set of linearly independent short vectors in a given lattice, are presumed difficult, even in the context of quantum computations. In order to assess the security of cryptographic schemes based on NTRU/RingSIS/etc, an essential question is whether the restriction to module lattices brings vulnerabilities. Putting aside small polynomial speed-ups relying on the field automorphisms (multiplication by x in our running example), the cryptanalytic state of the art is to view the modules as arbitrary lattices, i.e., forgetting the module structure.

LLL [LLL82] is the central algorithm to manipulate lattice bases. It takes as input a basis of a given lattice, progressively updates it, and eventually outputs another basis of the same lattice that is made of relatively short vectors. Its run-time is polynomial in the input bit-length. For cryptanalysis, one typically relies on BKZ [SE94] which extends this principle to find shorter vectors at a higher cost. Finding an analogue of LLL for module lattices has been an elusive goal for at least two decades, a difficulty being to even define what that would be. Informally, it should:

- work at the field level (in particular, it should not forget the module structure and view the module just as a lattice);
- it should find relatively short module pseudo-bases by progressively updating the input pseudo-basis;
- it should run in polynomial-time with respect to the module rank n and the bit-lengths of the norms of the input vectors and ideals.

The state of the art is far from these goals. Napias [Nap96] proposed such an algorithm for fields whose rings of integers are norm-Euclidean, i.e., Euclidean for the algebraic norm (in our running example, this restricts the applicability to $n \leq 4$). Fieker and Pohst [FP96] proposed a general-purpose algorithm. However, it was not proved to provide pseudo-bases consisting of short module vectors, and a cost analysis was provided only for free modules over totally real fields. Fieker [Fie97, p. 47] suggested to use rank-2 module reduction to achieve rank-m module reduction, but there was no follow-up on this approach. Gan, Ling and Mow [GLM09] described and analyzed an LLL algorithm for Gauss integers

[1] See `https://csrc.nist.gov/projects/post-quantum-cryptography`

(i.e., our running example instantiated to $n = 2$). Fieker and Stehlé [FS10] proposed to apply the LLL algorithm on the lattice corresponding to the module to find short vectors in polynomial time and reconstruct a short pseudo-basis afterwards. More recently, Kim and Lee [KL17] described such an LLL algorithm for biquadratic fields whose rings of integers are norm-Euclidean, and provided analyses for the shortness of the output and the run-time. They also proposed an extension to arbitrary norm-Euclidean rings, still with a run-time analysis but only conjecturing and experimentally supporting the output quality.

The rank-2 restriction already captures a fundamental obstacle. The LLL algorithm for 2-dimensional lattices (which is essentially Gauss' algorithm) is a succession of divide-and-swap steps. Given two vectors $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{Q}^2$, the 'division' consists in shortening $\mathbf{b}_2$ by an integer multiple of $\mathbf{b}_1$. This integer k is the quotient of the Euclidean division of $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle$ by $\|\mathbf{b}_1\|^2$. This leads to a vector $\mathbf{b}_2'$. If the latter is shorter than $\mathbf{b}_1$, then $\mathbf{b}_1$ and $\mathbf{b}_2'$ are swapped and a new iteration starts. Crucial to this procedure is the fact that if the projection of $\mathbf{b}_2$ orthogonally to $\mathbf{b}_1$ is very small compared to $\|\mathbf{b}_1\|$, then the division will provide a vector $\mathbf{b}_2'$ that is shorter than $\mathbf{b}_1$. When a swap cannot be made, it means that the projection of $\mathbf{b}_2$ orthogonally to $\mathbf{b}_1$ is not too small, and hence the basis is of good quality, i.e., somewhat orthogonal and hence made of somewhat short vectors. What provides the convergence to a short basis is the Euclideanity of $\mathbb{Z}$. This is why prior works focused on this setup. Put differently, the crucial property is the fact that the covering radius of the $\mathbb{Z}$ lattice is smaller than 1: this makes it possible to shorten a vector $\mathbf{b}_2$ whose projection is sufficiently small by an appropriate integer multiple such that $\mathbf{b}_2'$ becomes smaller than $\mathbf{b}_1$. When we extend to modules, the corresponding lattice becomes R, and its covering radius has no a priori reason to be smaller than 1 (for our running example, it is $\sqrt{n}/2$). Even if we allow an infinite amount of time to find an optimal $k \in R$, the resulting $\mathbf{b}_2 - k\mathbf{b}_1$ may still be longer than $\mathbf{b}_1$, even if $\mathbf{b}_2$ is in the K-span of $\mathbf{b}_1$. This leads us to the following question: does there exist a lattice L depending only on K such that being able to solve the Closest Vector Problem (CVP) with respect to L allows to find short bases of modules in K^2?

4.2 Contribution

The LLL algorithm for Euclidean lattices can be viewed as a way to leverage the ability of Gauss' algorithm to reduce 2-dimensional lattice bases, to reduce m-dimensional lattice bases for any $m \geq 2$. We propose extensions to modules of both Gauss' algorithm and of its LLL leveraging from 2 to n dimensions, hence providing a full-fledged framework for LLL-like reduction of module pseudo-bases.

Our first contribution in this chapter is to present a heuristic algorithm to find a very short non-zero vector in an arbitrary module in K^2, given access to a CVP oracle with respect to a lattice depending only on K. We obtain the following result for our running example (see Corollary 4.13 for a statement for arbitrary number fields).

Theorem 4.1 (Heuristic)**.** *There exists a sequence of lattices L_n and an algorithm $\mathcal{A}$ such that the following holds. Algorithm $\mathcal{A}$ takes as input a pseudo-basis of a rank-2 module $M \subset (\mathbb{Q}/(x^n+1))^2$ (with n a power of two), and outputs a vector $\mathbf{v} \in M \setminus \{0\}$ that is no more than $2^{(\log n)^{O(1)}}$ longer than a shortest non-zero vector of M. If given access to an oracle solving CVP in L_n, then it runs in quantum polynomial time. Finally, for any $\eta > 0$, the lattice L_n can be chosen of dimension $O(n^{2+\eta})$.*

The quantum component of the algorithm is the decomposition of an ideal as the product of a subset of fixed ideals and a principal ideal with a generator [BS16]. By relying on [BEF+17] instead, one can obtain a dequantized variant of Theorem 4.1 relying on more heuristics and in which the algorithm runs in $2^{\widetilde{O}(\sqrt{n})}$ classical time.

We insist that the result relies on heuristics. It relies on GRH and Heuristics 3.6, 3.7 and 3.12 presented in the previous chapter, as well as a new Heuristic 4.5, defined in Section 4.3.3. The new heuristic quantifies the distance to L_n of vectors in the real span of L_n that satisfy some properties. This heuristic is difficult to prove as the lattice L_n involves other lattices that are not very well understood (the log-unit lattice and the lattice of class group relations between ideals of small algebraic norms). We justify this heuristic by informal counting arguments and by some experiments supporting parts of these counting arguments.

Finally, we note that the dimension of L_n is essentially quadratic in the degree n of the field. This is much more than the lattice dimension n of R, but we do not know how to use a CVP oracle for R to

obtain such an algorithm to find short vectors in rank-2 modules. An alternative approach to obtain a similar reduction from finding short non-zero vectors in rank-2 modules to CVP with preprocessing would be as follows: to reach the goal, it suffices to find a short non-zero vector in a $(2n)$-dimensional lattice; by using the LLL algorithm and numerical approximations (see, e.g., [SMSV14]), it is possible to further assume that the bit-size of the inputs is polynomial in n; by Kannan's search-to-decision reduction for the shortest vector problem [Kan87], it suffices to obtain an algorithm that decides whether or not a lattice contains a non-zero vector of norm below 1; the latter task can be expressed as an instance of 3SAT, as the corresponding language belongs to NP; finally, 3SAT reduces to CVP with preprocessing [Mic01]. Overall, this gives an alternative to Theorem 4.1 without heuristics, but lattices L_n of much higher dimensions (which still grow polynomially in n).

As a second contribution, we describe an oracle-based algorithm which takes as input a pseudo-basis of a module $M \subset K^m$ over the ring of integers R of an arbitrary number field K, updates it progressively in a fashion similar to the LLL algorithm, and outputs a pseudo-basis of M. The first output vector is short, and the algorithm runs in time polynomial in m and the bit-lengths of the norms of the input vectors and ideals. It makes a polynomial number of calls to an oracle that finds short vectors in rank-2 modules. This oracle-based LLL-like algorithm for modules allows us to obtain the following result for power-of-two cyclotomic fields (see Theorem 4.22 for a general statement).

Theorem 4.2. *Let $K = \mathbb{Q}[x]/(x^n + 1)$ and $R = \mathbb{Z}[x]/(x^n + 1)$, for n a power of 2. There is a polynomial-time reduction from finding a $(2\gamma\sqrt{n})^{2m-1}$-approximation to a shortest non-zero vector in modules in K^m (with respect to the Euclidean norm inherited from mapping an element of K^m to the concatenation of its m coefficient vectors) to finding a γ-approximation to a shortest non-zero vector in modules in K^2.*

For example, if m is constant, then the reduction allows to obtain polynomial approximation factors in modules in K^m from polynomial approximation factors in modules in K^2.

Combining this reduction with the heuristic algorithm described above gives us a heuristic LLL algorithm for modules of rank m over R. This algorithm runs in quantum polynomial time, provided that it has access to an oracle solving CVP in a fixed lattice of dimension $\widetilde{O}(n^{2+\eta})$, for some $\eta > 0$ (again, in the case of power-of-two cyclotomic field).

4.2.1 Technical overview

One of the technical difficulties of extending LLL to modules is the fact that the absolute value $|\cdot|$ over $\mathbb{Q}$ has two canonical generalizations over K: the trace norm and the absolute value of the algebraic norm. Let $(\sigma_i)_{i\leq n}$ denote the embedding of K into $\mathbb{C}$. The trace norm and the absolute value of the algebraic norm of $x \in K$ are respectively defined as $(\sum_i |\sigma_i(x)|^2)^{1/2}$ and $\prod_i |\sigma_i(x)|$. When $K = \mathbb{Q}$, the only embedding is the identity map, and both the trace norm and the absolute value of the algebraic norm collapse to the absolute value. When the field degree is greater than 1, they do not collapse, and are convenient for diverse properties. For instance, the trace norm is convenient to measure smallness of a vector over K^m. A nice property is that the bit-size of an element of R is polynomially bounded in the bit-size of the trace norm (for a fixed field K). Oppositely, an element in R may have algebraic norm 1, but can have arbitrarily large bit-size. On the other hand, the algebraic norm is multiplicative, which interacts well with determinants. For example, the determinant of the lattice corresponding to a diagonal matrix over K is simply the product of the algebraic norms of the diagonal entries (up to a scalar depending only on the field K). When $K = \mathbb{Q}$, LLL relies on all these properties, that are conveniently satisfied by the absolute value.

Our algorithm for finding short non-zero vectors in rank-2 modules iterates divide-and-swap steps like 2-dimensional LLL (or Gauss' algorithm). The crucial component is the generalization of the Euclidean division, from $\mathbb{Z}$ to R. We are given $a \in K \setminus \{0\}$ and $b \in K$, and we would like to shorten b using R-multiples of a. In the context of $a \in \mathbb{Q} \setminus \{0\}$ and $b \in \mathbb{Q}$, a Euclidean division provides us with $u \in \mathbb{Z}$ such that $|b + ua| \leq |a|/2$. We would like to have an analogous division in R. However, the ring R may not be Euclidean. Moreover, the covering radius of the ring R (viewed as a lattice) can be larger than 1, and hence, in most cases, there will not even exist an element $u \in R$ such that $\|b + au\| \leq \|a\|$ (here $\|\cdot\|$ refers to the trace norm). In order to shorten b using a, we also allow b to be multiplied by some element $v \in R$. For this extension to be non-trivial (and useful), we require that v is not too large

(otherwise, one can always take $u = b$ and $v = -a$ for instance, if $a, b \in R$, and extend this approach for general $a, b \in K$). Hence, we are interested in finding u, v such that $\|ua + vb\| \leq \varepsilon \|a\|$ and $\|v\| \leq C$ for some $\varepsilon < 1$ and C to be determined later. Intuitively, if we allow for a large number of such multiples v (proportional to $1/\varepsilon$ and to the determinant of the lattice corresponding to R, i.e., the square root of the field discriminant), there should be one such v such that there exists $u \in R$ with $\|vb + au\| \leq \varepsilon \|a\|$. We do not know how to achieve the result with this heuristically optimal number of v's and use potentially larger v's. The astute reader will note that if we use such a v inside a divide-and-swap algorithm, we may end up computing short vectors in sub-modules of the input modules. We prevent this from happening by using the module Hermite Normal Form [BP91, Coh96, BFH17].

To find u, v such that $\|vb + au\|$ is small, we use the logarithm map Log over K. For this discussion, we do not need to explain how it is defined, but only that it "works" like the logarithm map log over $\mathbb{R}_{>0}$. In particular if $x \approx y$, then $\operatorname{Log} x \approx \operatorname{Log} y$. We would actually prefer to have the converse property, but it does not hold for the standard Log over K. In Subsection 4.3.1, we propose an extension $\overline{\operatorname{Log}}$ such that $\overline{\operatorname{Log}}\, x \approx \overline{\operatorname{Log}}\, y$ implies that $x \approx y$. In our context, this means that we want to find u, v such that $\overline{\operatorname{Log}}\, v - \overline{\operatorname{Log}}\, u \approx \overline{\operatorname{Log}}\, b - \overline{\operatorname{Log}}\, a$. To achieve this, we will essentially look for such u and v that are product combinations of fixed small elements in R. When applying the $\overline{\operatorname{Log}}$ function, the product combinations become integer combinations of the $\overline{\operatorname{Log}}$'s of the fixed elements. This gives us our CVP instance: the lattice is defined using the $\overline{\operatorname{Log}}$'s of the fixed elements and the target is defined using $\overline{\operatorname{Log}}(b) - \overline{\operatorname{Log}}(a)$. This description is only to provide intuition, as reality is more technical: we use the log-unit lattice and small-norm ideals rather than small-norm elements.

One advantage of using the $\overline{\operatorname{Log}}$ map is that the multiplicative structure of K is mapped to an additive structure, hence leading to a CVP instance. On the downside, one needs extreme closeness in the $\overline{\operatorname{Log}}$ space to obtain useful closeness in K (in this direction, we apply an exponential function). Put differently, we need the lattice to be very dense so that there is a lattice vector that is very close to the target vector. This is the fundamental reason why we end up with a large lattice dimension: we add a large number of $\overline{\operatorname{Log}}$'s of small-norm ideals to densify the lattice. This makes the analysis of the distance to the lattice quite cumbersome, as the Gaussian heuristic gives too crude estimates. For our running example, we have a lattice of dimension $\approx n^2$ and determinant ≈ 1, hence we would expect a 'random' target vector to be at distance $\approx n$ from the lattice. We argue for a distance of at most $\approx \sqrt{n}$ for 'specific' target vectors. Finally, we note that the lattice and its analysis share similarities with the Schnorr-Adleman lattice that Ajtai used to prove NP-hardness of SVP under randomized reductions [Ajt98, MG02] (but we do not know if there is a connection).

Recall that when extending the LLL algorithm to arbitrary number fields, one can choose between two generalizations of the absolute value: the trace norm or the absolute value of the algebraic norm. In our second contribution, i.e., the LLL algorithm for module lattices, we crucially rely on the algebraic norm. Indeed, the progress made by the LLL algorithm is measured by the so-called potential function, which is a product of determinants. As observed in prior works [FP96, KL17], using the algebraic norm allows for a direct generalization of this potential function to module lattices. What allowed us to go beyond norm-Euclidean number fields is the black-box handling of rank-2 modules. By not considering this difficult component, we can make do with the algebraic norm for the most important parts of the algorithm. The trace norm is still used to control the bit-sizes of the module pseudo-bases occurring during the algorithm, allowing to extend the so-called size-reduction process within LLL, but is not used to "make progress". The black-boxing of the rank-2 modules requires the introduction of a modified condition for deciding which 2-dimensional pseudo-basis to consider to "make progress" on the m-dimensional pseudo-basis being reduced. This condition is expressed as the ratio between 2-determinants, which is compatible with the exclusive use of the algebraic norm to measure progress. It involves the coefficient ideals, which was unnecessary in prior works handling norm-Euclidean fields, as for such fields, all modules can be generated by a basis instead of a pseudo-basis.

4.2.2 Impact

Recent works have showed that lattice problems restricted to ideals of some cyclotomic number fields can be quantumly solved faster than for arbitrary lattices, for some ranges of parameters [CDW17], and for all number fields with not too large discriminant, if allowing preprocessing that depends only on the field [PHS19] (see Chapter 3). Recall that ideal lattices are rank-1 module lattices. Our work

can be viewed as a step towards assessing the existence of such weaknesses for modules of larger rank, which are those that appear when trying to cryptanalyze cryptosystems based on the NTRU, RingSIS, RingLWE, ModuleSIS and ModuleLWE problems and their variants.

Similarly to [CDW17,PHS19], our results use CVP oracles for lattices defined in terms of the number field only (i.e., defined independently of the input module). In [CDW17, PHS19], the weaknesses of rank-1 modules stemmed from two properties of these CVP instances: the lattices had dimension quasi-linear in the log-discriminant (quasi-linear in the field degree, for our running example), and either the CVP instances were easy to solve [CDW17], or approximate solutions sufficed [PHS19] and one could rely on Laarhoven's CVP with preprocessing algorithm [Laa16]. In our case, we need (almost) exact solutions to CVP instances for which we could not find any efficient algorithm, and the invariant lattice has a dimension that is more than quadratic in the log-discriminant (in the field degree, for our running example). It is not ruled out that there could be efficient CVP algorithms for such lattices, maybe for some fields, but we do not have any lead to obtain them.

As explained earlier, CVP with preprocessing is known to be NP-complete, so there always exists a fixed lattice allowing to solve the shortest vector problem in lattices of a target dimension. However, the dimension of that fixed lattice grows as a high degree polynomial in the target dimension. The fact that we only need near-quadratic dimensions (when the log-discriminant is quasi-linear in the field degree) may be viewed as a hint that finding short non-zero vectors in rank-2 modules might be easier than finding short non-zero vectors in arbitrary lattices of the same dimension.

Finally, our first result shows the generality of rank-2 modules towards finding short vectors in rank-n modules for any $n \geq 2$. The reduction allows to stay in the realm of polynomial approximation factors (with respect to the field degree) for any constant n. This tends to back the conjecture that there might be a hardness gap between rank-1 and rank-2 modules, and then a smoother transition for higher rank modules.

4.3 Divide-and-swap algorithm for rank-2 modules

In this section, we describe a divide-and-swap algorithm for rank-2 modules, which mimic the 2-dimensional LLL algorithm (or Gauss' algorithm). In the first subsection below, we define a new logarithm function, which will be well suited for our objective. We then describe a lattice L that depends only on K and for which we will assume that we possess a CVP oracle. We also present our new heuristic assumption and give some justification for it. Then, we propose an algorithm whose objective is to act as a Euclidean algorithm, i.e., enabling us to shorten an element of $K_{\mathbb{R}}$ using R-multiples of another. Once we have this generalization of the Euclidean algorithm, we finally describe a divide-and-swap algorithm for rank-2 modules.

4.3.1 Extending the logarithm

In this section, we introduce an extension of the Log function that we used in the previous chapter. The difficulty with the usual Log function is that, for $a, b \in K_{\mathbb{R}}^{\times}$, the closeness between a and b is not necessarily implied by the closeness of $\operatorname{Log} a$ and $\operatorname{Log} b$. This is due to the fact that Log does not take into account the complex arguments of the entries of the canonical embeddings of a and b. However, we will need such a property to hold. For this purpose, we hence extend the Log function by also including the argument of the embeddings. For $x \in K_{\mathbb{R}}^{\times}$, we define

$$\overline{\operatorname{Log}}\, x := (\theta_1, \ldots, \theta_{r_1+r_2}, \log|\sigma_1(x)|, \ldots, \log|\sigma_n(x)|)^T,$$

where $\sigma_i(x) = |\sigma_i(x)| \cdot \mathrm{e}^{\mathrm{I}\theta_i}$ for all $i \leq r_1 + r_2$. The $\overline{\operatorname{Log}}$ function takes values in $(\pi\mathbb{Z}/2\pi\mathbb{Z})^{r_1} \times (\mathbb{R}/(2\pi\mathbb{Z}))^{r_2} \times \mathbb{R}^n$.

Lemma 4.3. *For $x, y \in K_{\mathbb{R}}^{\times}$, we have:*

$$\|x - y\|_\infty \leq \left(\mathrm{e}^{\sqrt{2}\|\overline{\operatorname{Log}}\, x - \overline{\operatorname{Log}}\, y\|_\infty} - 1\right) \cdot \min(\|x\|_\infty, \|y\|_\infty).$$

Observe that for $t \leq (\ln 2)/\sqrt{2}$, we have $\mathrm{e}^{\sqrt{2}t} - 1 \leq 2\sqrt{2}t$.

Proof. Let us write

$$\overline{\operatorname{Log}} x = (\theta_1, \dots, \theta_{r_1+r_2}, \log|\sigma_1(x)|, \dots, \log|\sigma_{r_1+r_2}(x)|)^T$$
$$\text{and } \overline{\operatorname{Log}} y = (\theta'_1, \dots, \theta'_{r_1+r_2}, \log|\sigma_1(y)|, \dots, \log|\sigma_{r_1+r_2}(y)|)^T.$$

Let $i \leq r_1 + r_2$ and define $z_x = \log|\sigma_i(x)| + \mathrm{I}\theta_i$ and $z_y = \log|\sigma_i(y)| + \mathrm{I}\theta'_i$. By definition of $\overline{\operatorname{Log}}$, we have $\sigma_i(x) = \mathrm{e}^{z_x}$ and $\sigma_i(y) = \mathrm{e}^{z_y}$. Therefore, we can write

$$\frac{|\sigma_i(x) - \sigma_i(y)|}{|\sigma_i(x)|} = |1 - \mathrm{e}^{z_y - z_x}| = \Big|\sum_{k\geq 1} \frac{(z_y - z_x)^k}{k!}\Big| \leq \sum_{k\geq 1} \frac{|z_y - z_x|^k}{k!} = \mathrm{e}^{|z_y - z_x|} - 1.$$

As $|z_x - z_y| \leq \sqrt{2}\|\overline{\operatorname{Log}} x - \overline{\operatorname{Log}} y\|_\infty$, we derive that

$$|\sigma_i(x) - \sigma_i(y)| \leq \|x\|_\infty \cdot \left(\mathrm{e}^{\sqrt{2}\|\overline{\operatorname{Log}} x - \overline{\operatorname{Log}} y\|_\infty} - 1\right).$$

Note that this holds for all $i \leq r_1 + r_2$ and that we could as well have used $\|y\|_\infty$ rather than $\|x\|_\infty$. □

4.3.2 The lattice L

Let $r = \mathrm{poly}(n)$ and $\beta > 0$ be some parameter to be chosen later. Let Λ denote the log-unit lattice. Let $\mathfrak{B}_0 = \{\mathfrak{p}_1, \dots, \mathfrak{p}_{r_0}\}$ be a set of cardinality $r_0 \leq \log h_K$ of prime ideals generating $\mathcal{C}l_K$, with algebraic norms $\leq 2^{\delta_0}$, with $\delta_0 = O(\log\log|\Delta|)$. We will also consider another set $\mathfrak{B} = \{\mathfrak{q}_1, \dots, \mathfrak{q}_r\}$ of cardinality r, containing prime ideals (not in $\mathfrak{B}_0$) of norms $\leq 2^\delta$, for some parameters r and $\delta \leq \delta_0$ to be chosen later. We also ask that among these ideals $\mathfrak{q}_j$, at least half of them have an algebraic norm $\geq \sqrt{2^\delta}$. Because we want r such ideals, we should make sure that the number of prime ideals of norm bounded by 2^δ in R is larger than r. This is asymptotically satisfied if $(\log|\Delta|)^2 \leq r \leq 2^\delta/(\log|\Delta|)^\varepsilon$ for some $\varepsilon > 0$ (by Theorem 2.13 and Corollary 2.14). The constraint that at least $r/2$ ideals should have norm larger than $\sqrt{2^\delta}$ is not very limiting, as we expect that roughly $2^\delta - \sqrt{2^\delta} \geq r - \sqrt{r}$ ideals should have algebraic norm between $\sqrt{2^\delta}$ and 2^δ (forgetting about the $\mathrm{poly}(\delta)$ terms).

We now define L as the lattice of dimension $\nu = 2(r_1 + r_2) + r_0 + r - 1$ spanned by the columns of the following basis matrix:

$$\mathbf{B}_L := \begin{array}{|c|c|c|c|l}
\multicolumn{2}{c}{\overleftarrow{\qquad 2(r_1+r_2)-1 \qquad}} & \multicolumn{2}{c}{\overleftarrow{\qquad\qquad r_0 + r \qquad\qquad}} & \\
\multicolumn{2}{|c|}{0} & \multicolumn{2}{c|}{\beta \cdot a_{g_{2(r_1+r_2)}} \cdots \beta \cdot a_{g_\nu}} & \updownarrow 1 \\
\hline
\beta \cdot 2\pi & \multicolumn{3}{c|}{} & \\
\quad \ddots & \multicolumn{3}{c|}{\beta \cdot \theta_{g_{r_1+r_2+1}} \quad \cdots \quad \beta \cdot \theta_{g_\nu}} & r_1 + r_2 \\
\beta \cdot 2\pi & \multicolumn{3}{c|}{} & \\
\hline
0 & \beta \cdot \mathbf{B}_\Lambda & \multicolumn{2}{c|}{\beta \cdot \mathbf{h}_{g_{2(r_1+r_2)}} \cdots \beta \cdot \mathbf{h}_{g_\nu}} & r_1 + r_2 - 1 \\
\hline
\multicolumn{2}{|c|}{} & \mathbf{B}_{\Lambda_0} & \mathbf{w}_{\nu-r+1} \cdots \mathbf{w}_\nu & r_0 \\
\hline
\multicolumn{2}{|c|}{0} & 0 & 1 \quad \ddots \quad 1 & r
\end{array}$$

where

- $\mathbf{B}_\Lambda$ is a basis of Λ, and we let $(\mathbf{h}_i)_{r_1+r_2<i<2(r_1+r_2)}$ denote its columns;

- $\mathbf{B}_{\Lambda_0}$ is a basis of the lattice $\Lambda_0 := \{(x_j)_j \in \mathbb{Z}^{r_0} : \prod_j \mathfrak{p}_j^{x_j} \text{ is principal}\}$, and we let $(\mathbf{w}_i)_{2(r_1+r_2)\leq i\leq \nu-r}$ denote its columns;
- for any $g \in K$, we have $a_g = (\log|\mathcal{N}(g)|)/\sqrt{n}$;
- for any $g \in K$, the vector θ_g consists of the first $r_1 + r_2$ entries of $\overline{\log}(g)$;
- for any $g \in K$, we have $\mathbf{h}_g = i_{H\cap E}(\Pi_H(\log(g)))$, where Π_H is the orthogonal projection on H and $i_{H\cap E}$ is an isometry mapping $H \cap E$ to $\mathbb{R}^{r_1+r_2-1}$;
- for any $i > r_1 + r_2$, if we parse the bottom $r_0 + r$ coordinates of the i-th column vector as $(w_{i,1}, \ldots, w_{i,r_0}, w'_{i,1}, \ldots, w'_{i,r})$, then we have that $\langle g_i\rangle = \prod_j \mathfrak{p}_j^{w_{ij}} \cdot \prod_j \mathfrak{q}_j^{w'_{ij}}$;
- the g_i's for $i > r_1 + r_2$ are in K and, among them, $g_{r_1+r_2+1}, \ldots g_{2(r_1+r_2)-1}$ are the units of R corresponding to the columns of $\mathbf{B}_\Lambda$.

We now list a few properties satisfied by vectors in this lattice.

Lemma 4.4. *For every vector $(\beta a\|\beta\theta\|\beta\mathbf{h}\|\mathbf{w}\|\mathbf{w}') \in L$ (with blocks of dimensions $1, r_1+r_2, r_1+r_2-1, r_0$ and r), there exists $g \in K \setminus \{0\}$ with*

- $a = (\log|\mathcal{N}(g)|)/\sqrt{n}$;
- $\overline{\log}(g) = (\theta'\|\log(g))$ *with* $\theta' = \theta \bmod 2\pi$;
- $\mathbf{h} = i_{H\cap E}(\Pi_H(\log(g)))$;
- $\langle g\rangle = \prod_j \mathfrak{p}_j^{w_j} \prod_j \mathfrak{q}_j^{w'_j}$.

Further, we have that $\|\log(g)\|_2 = \|(a, \mathbf{h})\|_2$.

Proof. Note that the first claim holds for the vectors of the input basis (with $g = 1$ for the first $r_1 + r_2$ vectors). The property is preserved by vector addition (resp. subtraction), as can be seen by multiplying (resp. dividing) the corresponding g's. Hence it holds for all vectors of the lattice.

For the last statement, observe that $\log(g) = \frac{\log|\mathcal{N}(g)|}{n}\mathbf{1} + \Pi_H(\log(g))$. Hence, by orthogonality, we have:

$$\|\log(g)\|_2^2 = \left(\frac{\log|\mathcal{N}(g)|}{n}\right)^2 \cdot n + \|\Pi_H(\log(g))\|_2^2 = \|(a, \mathbf{h})\|_2^2.$$

This completes the proof. □

4.3.3 On the distance of relevant vectors to the lattice

In this section, we make a heuristic assumption on the distance between target vectors of a specific form and the lattice L defined in the previous section. This heuristic is backed with a counting argument. As L is not full rank, we only consider target vectors $\mathbf{t}$ lying in the span of L. Also, as $\mathbf{B}_L$ contains the identity matrix in its bottom right corner, we cannot hope to have a covering radius that is much smaller than $\sqrt{r}$. In our case, the lattice dimension ν will be of the order of r, but in our application we will need a vector of L much closer to $\mathbf{t}$ than $\sqrt{r} \approx \sqrt{\nu}$. In order to go below this value, we only consider target vectors $\mathbf{t}$ whose last r coordinates are almost 0.

Heuristic 4.5. *Assume that there exist some integer $B \leq r$ such that $B \geq 100 \cdot (\log h_K) \cdot \delta_0/\delta$ and that*

$$\alpha_0 := \sqrt{2\pi}\left(\left(\frac{2B}{r^{0.96}}\right)^B \cdot \delta B(\det\Lambda)h_K\right)^{1/n} \leq \frac{\ln n}{12n^2}.$$

Assume that the scaling parameter β in $\mathbf{B}_L$ is set to $\frac{1}{\alpha_0}\sqrt{\frac{0.01\cdot B}{2n}}$. Then for any $\mathbf{t} \in \mathrm{Span}(L)$ with its r last coordinates $\mathbf{w}'_t$ satisfying $\|\mathbf{w}'_t\|_2 \leq 0.01 \cdot B/\sqrt{r}$, we have dist$(\mathbf{t}, L) \leq \sqrt{1.05 \cdot B}$.

Discussion about Heuristic 4.5. We provide below a counting argument to justify Heuristic 4.5. We consider the following set of vectors of L, parametrized by $B \leq r$, which we view as candidates for very close vectors to such target vectors:

$$S_B := \{\mathbf{s} = (\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : \mathbf{w}'_s \in \{-1, 0, 1\}^r \wedge \|\mathbf{w}'_s\|_1 = B\}.$$

We argue that there is a vector in S_B that is very close to $\mathbf{t}$. We are going to examine the vectors $\mathbf{s} \in S_B$ such that $\mathbf{s} - \mathbf{t}$ is reduced modulo L. Let us write $\mathbf{t} = (\beta a_t \| \beta\theta_t \| \beta \mathbf{h}_t \| \mathbf{w}_t \| \mathbf{w}'_t)$. We define:

$$\begin{aligned} S^{(1)}_{B,\mathbf{t}} := \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : \; & \mathbf{w}'_s \in \{-1, 0, 1\}^r \wedge \|\mathbf{w}'_s\|_1 = B, \\ & \mathbf{w}_s - \lfloor \mathbf{w}_t \rceil \in \mathcal{V}(\Lambda_0), \\ & \mathbf{h}_t - \mathbf{h}_s \in \mathcal{V}(\Lambda), \\ & \theta_t - \theta_s \text{ is reduced mod } 2\pi\}, \end{aligned}$$

where the notation $\mathcal{V}$ refers to the Voronoi cell (i.e., the set of points which are closer to 0 than to any other point of the lattice). The choice of $\mathbf{w}'_s$ fully determines $\mathbf{s} \in S^{(1)}_{B,\mathbf{t}}$, which gives the bound $|S^{(1)}_{B,\mathbf{t}}| = 2^B \cdot \binom{r}{B} \geq (2r/B)^B$.

We consider the following subset of $S^{(1)}_{B,\mathbf{t}}$:

$$S^{(2)}_{B,\mathbf{t}} = S^{(1)}_{B,\mathbf{t}} \cap \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : \mathbf{w}_s = \lfloor \mathbf{w}_t \rceil\}.$$

We heuristically assume that when we sample a uniform vector in $S^{(1)}_{B,\mathbf{t}}$, the components $\mathbf{w}_s$ of the vectors $\mathbf{s} \in S^{(1)}_{B,\mathbf{t}}$ are uniformly distributed modulo Λ_0. Then the proportion of those for which $\mathbf{w}_s = \lfloor \mathbf{w}_t \rceil \bmod \Lambda_0$ is $1/\det(\Lambda_0) = 1/h_K$. Hence, we expect that $|S^{(2)}_{B,\mathbf{t}}| \approx |S^{(1)}_{B,\mathbf{t}}|/h_K$.

We consider the following subset of $S^{(2)}_{B,\mathbf{t}}$, parametrized by $\alpha < (\ln n)/(12n^2)$:

$$S^{(3)}_{B,\alpha,\mathbf{t}} = S^{(2)}_{B,\mathbf{t}} \cap \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : \|(\theta_s \| \mathbf{h}_s) - (\theta_t \| \mathbf{h}_t)\|_\infty \leq \alpha\}.$$

We heuristically assume that when we sample a uniform vector in $S^{(2)}_{B,\mathbf{t}}$, the components $(\theta_s, \mathbf{h}_s)$ are uniformly distributed modulo $2\pi\mathbb{Z}^{r_1+r_2} \times \Lambda$. Observe that the first r_1 coordinates of θ_s (corresponding to real embeddings) are either 0 or π. Hence, the probability that $\theta_s = \theta_t$ on these coordinates is 2^{-r_1}. Once these first r_1 coordinates are fixed, the remaining coordinates of $(\theta_s, \mathbf{h}_s)$ have no a priori reason to be bound to a sublattice of $2\pi\mathbb{Z}^{r_2} \times \Lambda$ and we heuristically assume them to be uniformly distributed in $\mathbb{R}^{r_1+2r_2-1}/(2\pi\mathbb{Z}^{r_2} \times \Lambda)$. Overall, the probability that a vector $\mathbf{s} \in S^{(2)}_{B,\mathbf{t}}$ satisfies $\|(\theta_s, \mathbf{h}_s) - (\theta_t, \mathbf{h}_t)\|_\infty \leq \alpha$ is $\approx \frac{\alpha^{r_1+2r_2-1}}{2^{r_1} \cdot (2\pi)^{r_2} \cdot \det(\Lambda)}$. Here, we used the fact that α is smaller than $\lambda_1(2\pi\mathbb{Z}^{r_2} \times \Lambda)/2$ (recall from preliminaries that $\lambda_1(\Lambda) \geq (\ln n)/(6n)$). We conclude that

$$|S^{(3)}_{B,\alpha,\mathbf{t}}| \approx |S^{(2)}_{B,\mathbf{t}}| \frac{\alpha^{r_1+2r_2-1}}{2^{r_1} \cdot (2\pi)^{r_2} \cdot \det(\Lambda)} \geq |S^{(2)}_{B,\mathbf{t}}| \frac{\alpha^{n-1}}{(2\pi)^{n/2} \cdot \det(\Lambda)}.$$

Finally, we consider the following subset of $S^{(3)}_{B,\alpha,\mathbf{t}}$:

$$S^{(4)}_{B,\alpha,\mathbf{t}} = S^{(3)}_{B,\alpha,\mathbf{t}} \cap \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : |a_s - a_t| \leq \alpha\}.$$

We want a lower bound for $|S^{(4)}_{B,\alpha,\mathbf{t}}|/|S^{(3)}_{B,\alpha,\mathbf{t}}|$. Take $\mathbf{s} \in S^{(3)}_{B,\alpha,\mathbf{t}}$. As $\mathbf{t} \in \mathrm{Span}(L)$, we have:

$$\sqrt{n}(a_t - a_s) = \sum_j (\mathbf{w}_{t,j} - \lfloor \mathbf{w}_{t,j} \rceil) \log \mathcal{N}(\mathfrak{p}_j) + \sum_j \mathbf{w}'_{t,j} \log \mathcal{N}(\mathfrak{q}_j) - \sum_j \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j),$$

where the $\mathbf{w}_{t,j}$ (respectively $\mathbf{w}'_{t,j}$ and $\mathbf{w}'_{s,j}$) are the coordinates of $\mathbf{w}_t$ (respectively $\mathbf{w}'_t$ and $\mathbf{w}'_s$). We define $b_t = \sum_j (\mathbf{w}_{t,j} - \lfloor \mathbf{w}_{t,j} \rceil) \log \mathcal{N}(\mathfrak{p}_j) + \sum_j \mathbf{w}'_{t,j} \log \mathcal{N}(\mathfrak{q}_j)$, which depends only on $\mathbf{t}$. We would like to have a lower bound on the proportion of vectors $\mathbf{s} \in S^{(3)}_{B,\alpha,\mathbf{t}}$ such that $|\sum_j \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j) - b_t| \leq \sqrt{n}\alpha$.

In other words, we would like a lower bound on the probability that $|\sum_j \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j) - b_t| \leq \sqrt{n}\alpha$, when $\mathbf{s}$ is chosen uniformly at random in $S^{(3)}_{B,\alpha,\mathbf{t}}$. In order to simplify the estimation, we will assume in the following that when $\mathbf{s}$ is chosen uniformly at random in $S^{(3)}_{B,\alpha,\mathbf{t}}$, then the vector $\mathbf{w}'_s$ is sampled uniformly in $\{-1,0,1\}^r$, with B non-zero coefficients (this is an over-simplification, as we are already restricted to $S^{(3)}_{B,\alpha,\mathbf{t}}$). Let us write Y_j the random variables $\mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j)$ for all $1 \leq j \leq r$.

First, let us compute an upper bound on $|b_t|$ (the random variable $|\sum_j Y_j|$ being bounded by δB, if $|b_t| > \delta B + \alpha\sqrt{n}$, then the event would be empty). We have

$$|b_t| \leq \sum_j |\mathbf{w}_{t,j} - \lfloor \mathbf{w}_{t,j} \rceil| \cdot \delta_0 + \sum_j |\mathbf{w}'_{t,j}| \cdot \delta \leq \delta_0 \cdot r_0 + \delta \cdot \|\mathbf{w}'_t\|_1 \leq 0.02 \cdot \delta B,$$

using the assumptions on B and $\|\mathbf{w}'_t\|_1 \leq \sqrt{r} \cdot \|\mathbf{w}'_t\|$.

The element b_t is hence small enough to be reached by the variable $\sum_j Y_j$. However, it will be in the tail of the distribution (the sum of B independent variables bounded by δ has standard deviation at most $\sqrt{B} \cdot \delta$, which is asymptotically smaller than $|b_t|$). This makes the probability very small and difficult to estimate. In order to circumvent this difficulty, we will re-center our target b_t. To do so, we will condition our probability on the event that the first $0.04 \cdot B$ variables Y_j are such that their sum is very close to b_t.

More formally, recall that we chose our ideals so that at least $B/2$ of the ideals $\mathfrak{q}_j$ have norms at least $2^{\delta/2}$. Let us then sort the ideals $\mathfrak{q}_j$ by decreasing algebraic norm. By assumption, we know that the first $B/2$ ideals satisfy $\delta/2 \leq \log \mathcal{N}(\mathfrak{q}_j) \leq \delta$ for $j \leq B/2$. Now, because $|b_t| \leq 0.02 \cdot \delta B$, we know that there exists a choice of $\mathbf{w}'_{s,1}, \ldots, \mathbf{w}'_{s,\lceil 0.04 \cdot B \rceil} \in \{-1, 1\}$ such that $|\sum_{j \leq 0.04 \cdot B} Y_j - b_t| \leq \delta$. Indeed, we can for instance choose the signs of the first $\mathbf{w}'_{s,j}$ to be the same as the sign of b_t, until we reach b_t (we will reach it because it is smaller that $0.02 \cdot \delta B$ and we can sum up to $0.04 \cdot B$ elements of absolute value at least $\delta/2$). Once we have reached b_t, we can choose the following signs to be negative if the sum is larger than b_t and positive if the sum is smaller than b_t, this ensures that we will never be at distance more than δ from b_t. Let us define $b'_t := b_t - \sum_{j \leq 0.04 \cdot B} Y_j$. We have seen that we can force the values of the first $\mathbf{w}'_{s,j}$ such that $|b'_t| \leq \delta$. We are then left with $0.96 \cdot B$ non-zero values $\mathbf{w}_{s,j}$ to choose among the $r - 0.04 \cdot B$ remaining ideals, but our target vector is now b'_t, which is much smaller than b_t (in particular, it is asymptotically smaller than the standard deviation). Overall, we obtain the following equations, where the probabilities are taken over the choice of the $\mathbf{w}'_{s,j}$ (chosen in $\{-1,0,1\}^r$, uniformly with B non-zero coefficients).

$$\begin{aligned}
&\Pr(|\sum_j Y_j - b_t| \leq \sqrt{n}\alpha) \\
&\geq \Pr\big(|\sum_j Y_j - b_t| \leq \sqrt{n}\alpha \text{ and } |b_t - \sum_{j \leq 0.04 \cdot B} Y_j| \leq \delta \text{ and } \mathbf{w}'_{s,j} = \pm 1 \ \forall j \leq 0.04 \cdot B\big) \\
&= \Pr\big(|\sum_j Y_j - b_t| \leq \sqrt{n}\alpha \ \big| \ |b_t - \sum_{j \leq 0.04 \cdot B} Y_j| \leq \delta \text{ and } \mathbf{w}'_{s,j} = \pm 1 \ \forall j \leq 0.04 \cdot B\big) \\
&\quad \cdot \Pr\big(|b_t - \sum_{j \leq 0.04 \cdot B} Y_j| \leq \delta \text{ and } \mathbf{w}'_{s,j} = \pm 1 \ \forall j \leq 0.04 \cdot B\big).
\end{aligned}$$

We have seen that there exists at least one choice of the first $0.04 \cdot B$ elements $\mathbf{w}'_{s,j}$ such that $|b_t - \sum_{j \leq 0.04 \cdot B} Y_j| \leq \delta$. Hence, by counting the number of possible choices for the remaining $0.96 \cdot B$

non-zero elements $\mathbf{w}'_{s,j}$, we obtain that

$$\begin{aligned}
&\Pr\left(|b_t - \sum_{j\leq 0.04\cdot B} Y_j| \leq \delta \text{ and } \mathbf{w}'_{s,j} = \pm 1 \ \forall j \leq 0.04\cdot B\right) \\
&\geq \frac{2^{0.96\cdot B}\cdot \binom{r-0.04\cdot B}{0.96\cdot B}}{2^B \cdot \binom{r}{B}} \\
&\geq 2^{-0.04\cdot B}\cdot \left(\frac{r-0.04\cdot B}{0.96\cdot B}\right)^{0.96\cdot B} \cdot \left(\frac{B}{\mathrm{e}\cdot r}\right)^B \\
&\geq \frac{(r-0.04\cdot B)^{0.96\cdot B}}{\mathrm{e}^B \cdot 2^{0.04\cdot B}\cdot r^B} \\
&\geq \frac{(0.96\cdot r)^{0.96\cdot B}}{\mathrm{e}^B \cdot 2^{0.04\cdot B}\cdot r^B} \geq \frac{0.344^B}{r^{0.04\cdot B}},
\end{aligned}$$

where we used the fact that for any $B \leq r$, it holds that $(r/B)^B \leq \binom{r}{B} \leq (\mathrm{e}\cdot r/B)^B$.

It now remains to compute a lower bound on the probability that $|\sum_j Y_j - b_t| \leq \alpha\sqrt{n}$ conditioned on the fact that the first $\mathbf{w}'_{s,j}$ are chosen so that $|\sum_{j\leq 0.04\cdot B} Y_j - b_t| \leq \delta$. As mentioned above, this probability is equal to $\Pr(|\sum_{j>0.04\cdot B} Y_j - b'_t| \leq \alpha\sqrt{n})$, where $|b'_t| \leq \delta$ and the $\mathbf{w}'_{s,j}$ are chosen uniformly at random in $\{-1,0,1\}^{r-0.04\cdot B}$, with $0.96\cdot B$ non-zero coefficients. Because our target is now close to the center of the distribution, and $\sum_{j>0.04\cdot B} Y_j$ has a bell shape with standard deviation roughly $\sqrt{0.96\cdot B}\cdot\delta$ which is much larger than b'_t, we will assume that

$$\Pr(|\sum_{j>0.04\cdot B} Y_j - b'_t| \leq \alpha\sqrt{n}) \geq \frac{\alpha\sqrt{n}}{B\delta},$$

where the lower bound is the probability we would have obtained if the random variable $\sum_{j>0.04\cdot B} Y_j$ was uniformly distributed in $[-B\delta, B\delta]$. This assumption is justified by the fact that the random variable $\sum_{j>0.04\cdot B} Y_j$ has a bell shape, and b'_t is close to the center of the bell, hence the probability to be close to b'_t should be larger in the bell-shape case than in the uniform case (over $[-0.96\cdot B\delta, 0.96\cdot B\delta] \subset [-B\delta, B\delta]$). This lower bound on the probability is backed with numerical experiments, described in Section 4.3.3.1 below.

Overall, we obtain the following lower bound.

$$\Pr(|\sum_j Y_j - b_t| \leq \sqrt{n}\alpha) \geq \frac{0.344^B}{r^{0.04\cdot B}}\cdot\frac{\alpha\sqrt{n}}{B\delta}.$$

We finally obtain that

$$\begin{aligned}
|S^{(4)}_{B,\alpha,\mathbf{t}}| &\geq \frac{0.344^B\cdot\alpha\sqrt{n}}{\delta B\cdot r^{0.04\cdot B}}\cdot\frac{\alpha^{n-1}}{(2\pi)^{n/2}\cdot\det(\Lambda)}\cdot\frac{1}{h_K}\cdot\left(\frac{2r}{B}\right)^B \\
&\geq \left(\frac{\alpha}{\sqrt{2\pi}}\right)^n \frac{1}{\delta B\cdot\det(\Lambda)\cdot h_K}\left(\frac{0.344\cdot 2r}{B\cdot r^{0.04}}\right)^B \\
&\geq \left(\frac{\alpha}{\sqrt{2\pi}}\right)^n \frac{1}{\delta B\cdot\det(\Lambda)\cdot h_K}\left(\frac{r^{0.96}}{2B}\right)^B.
\end{aligned}$$

When the above is ≥ 1, we expect that there exists $\mathbf{s} \in S^{(4)}_{B,\alpha,\mathbf{t}}$. If that is the case, then we have

$$\|\mathbf{s}-\mathbf{t}\|^2 \leq (\beta\cdot\sqrt{2n}\cdot\alpha)^2 + r_0 + \|\mathbf{w}'_t - \mathbf{w}'_s\|^2.$$

By condition on B, we know that $r_0 \leq 0.01\cdot B$. Also, by choice of $\mathbf{w}'_t$ (and using the fact that $r \geq B$), we have that $\|\mathbf{w}'_t - \mathbf{w}'_s\|^2 \leq (\sqrt{B} + 0.01\cdot\sqrt{B})^2 \leq 1.03\cdot B$. Finally, choosing α minimal provides the result.

4.3.3.1 Numerical experiments

In this section, we provide some experimental results justifying the above assumption that

$$\Pr(|\sum_{j>0.04\cdot B} \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j) - b'_t| \leq \alpha\sqrt{n}) \geq \frac{\alpha\sqrt{n}}{B\delta},$$

for any target b'_t satisfying $|b'_t| \leq \delta$. Let us write $p_\ell = \Pr(|\sum_{j>0.04\cdot B} \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j) - b'_t| \leq \ell)$. We first checked that for a fixed number field and choice of B and δ, this probability is proportional to ℓ (for small ℓ's). To do so, we chose a random target b'_t uniformly in $[-\delta, \delta]$, and we computed the empirical probability p_ℓ for different values of ℓ. This empirical probability was computed by sampling 500 000 points according to the distribution $\sum_j \mathbf{w}'_{s,j} \log \mathcal{N}(\mathfrak{q}_j)$, and counting the number of them falling at distance at most ℓ of b'_t. We computed this empirical probability for different values of ℓ, ranging in $[0.001, 0.01]$, $[0.01, 0.1]$ and $[0.1, 1]$ (to check that the proportionality was still valid for different orders of magnitude of ℓ). The empirical probabilities are plotted in Figures 4.1, 4.2 and 4.3 for a power of two cyclotomic field, a cyclotomic field which is not a prime power, and a "random" number field (the coefficients of the defining polynomial being chosen uniformly at random between -4 and 4). One can observe that the probabilities are indeed proportional the length of the interval ℓ, and that the proportionality factor is roughly the same for the three different ranges of ℓ.

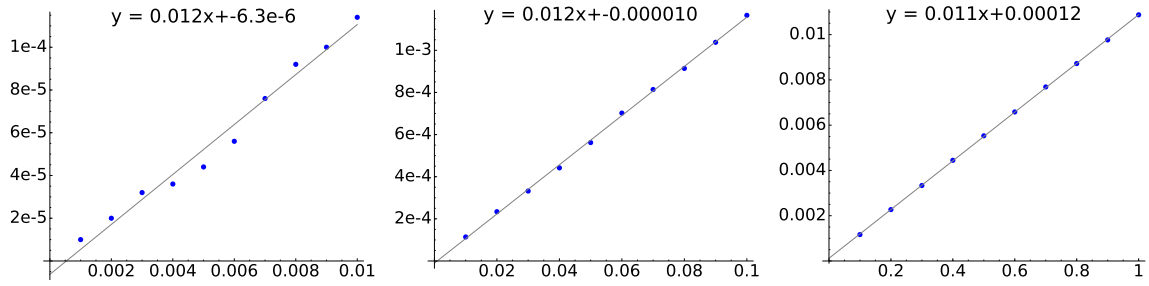

Figure 4.1: Empirical probability p_ℓ as a function of ℓ in a cyclotomic field of conductor 64 (degree 32)

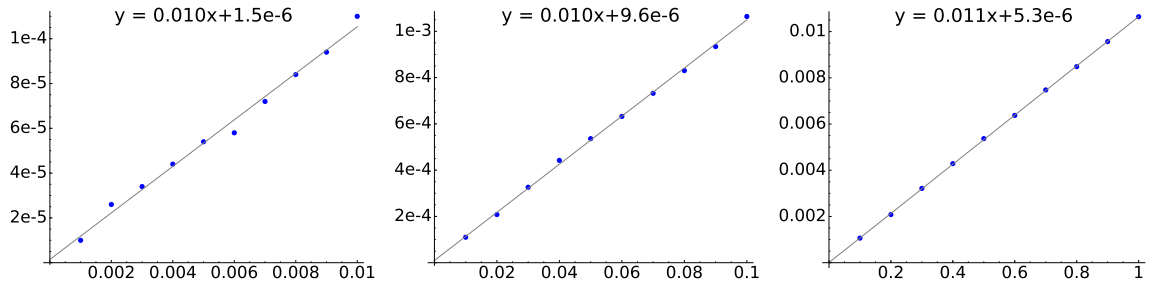

Figure 4.2: Empirical probability p_ℓ as a function of ℓ in a cyclotomic field of conductor 100 (degree 40)

Once we are convinced that the probability p_ℓ is indeed of the form $p_\ell = c \cdot \ell$, it remains to check the asymptotic behavior of the proportionality factor c, when n tends to infinity (the choice of n determines the choice of B and δ). We hence computed the proportionality factor c empirically for different cyclotomic number fields of increasing degree. The value of B was set as $B = n$ and δ was set as $\delta = n^2 \log(n^2)$ (these will roughly be the values we choose later in our algorithm). The number of points used to compute the empirical probabilities p_ℓ was set to 10 000 and we chose ℓ ranging in $[0.1, 1]$ to compute the slope.

Instead of checking that $c \geq \frac{1}{B\delta}$, we checked that $1/c \leq B\delta$. The values of $1/c$ and $B\delta$ are given in Table 4.1. One can observe that $1/c$ is indeed smaller than $B\delta$. We also plotted the values of $1/c$ as a function of $B\delta$ in Figure 4.4. We observe that the value $1/c$ seems to increase linearly in $B\delta$, with a slope smaller than 1, hence it seems that the inequality $1/c \leq B\delta$ should still hold asymptotically.

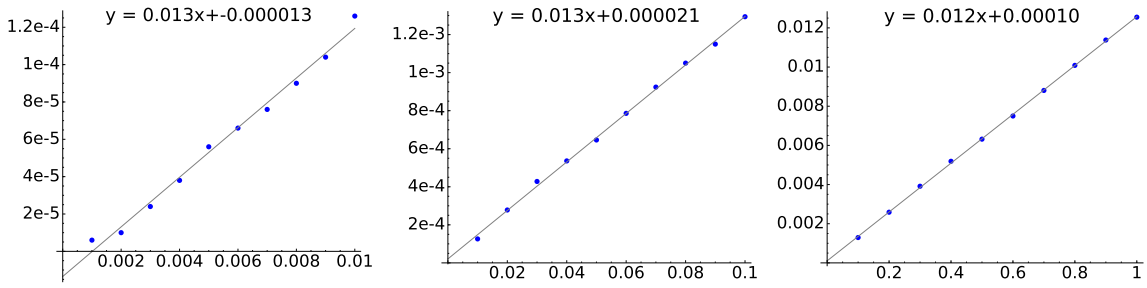

Figure 4.3: Empirical probability p_ℓ as a function of ℓ in a "random" number field of degree 32

n	8	16	20	24	32	36	40	44	48	54	56	60	64	66	70
m	16	32	44	35	64	57	100	69	65	81	87	99	128	67	71
$B \cdot \delta$	64	168	225	284	409	475	541	609	678	783	819	891	964	1000	1074
$1/c$	33	87	39	93	67	124	97	116	99	113	142	142	162	159	189

Table 4.1: Empirical values of $1/c$ for different cyclotomic number fields of conductor m and degree n

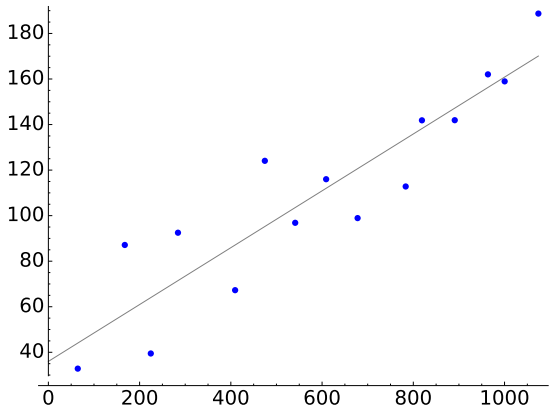

Figure 4.4: Evolution of $1/c$ (computed empirically) as a function of $B\delta$

4.3.4 A "Euclidean division" over R

We will need the following technical observation that, given $a, b \in K_{\mathbb{R}}$, it is possible to add a small multiple ka of a to b to ensure that $\mathcal{N}(b + ka) \geq \mathcal{N}(a)$.

Lemma 4.6. *For any $a \in K_{\mathbb{R}}^{\times}$ and $b \in K_{\mathbb{R}}$, there exists $k \in [-n, n] \cap \mathbb{Z}$ such that $|\mathcal{N}(b+ka)| \geq |\mathcal{N}(a)|$.*

Proof. We know that $|\mathcal{N}(a)| = \prod_i |\sigma_i(a)|$ so it suffices to find k such that $|\sigma_i(b + ak)| \geq |\sigma_i(a)|$ holds for all $i \leq n$. Now, the condition $|\sigma_i(b + ak)| < |\sigma_i(a)|$ is equivalent to $|\sigma_i(b/a) + k| < 1$. As a complex plane open circle of radius 1 contains at most two integers, we deduce that for each $i \leq n$, there are at most two integers k such that $|\sigma_i(b + ak)| < |\sigma_i(a)|$. Because the set $[-n, n] \cap \mathbb{Z}$ contains $2n + 1$ different values of k, then at least one of these should satisfy $|\sigma_i(b + ak)| \geq |\sigma_i(a)|$ for all i. □

Note that an integer k such as in Lemma 4.6 can be found efficiently by exhaustive search.

We can now describe our "Euclidean division" algorithm over R. Our algorithm takes as input a fractional ideal $\mathfrak{a}$ and two elements $a, b \in K_{\mathbb{R}}$, and outputs a pair $(u, v) \in R \times \mathfrak{a}$. The first five steps of this algorithm aim at obtaining, for any input (a, b), a replacement (a_1, b_1) that satisfies some conditions. Namely, we would like a_1 to be balanced, i.e., $\|a_1\|$ should not be significantly more than $\mathcal{N}(a_1)^{1/n}$. We also would like b_1 to be not much larger that a_1 and $\mathcal{N}(a_1/b_1)$ to be close to 1. These conditions are obtained by multiplying the element a by an appropriate element of R, and removing a multiple of a from b. Note that we require that the output element v should not be too large. As b is

not multiplied by anything, these normalization steps will not impact this output property. After these first five steps, the core of the algorithm begins. It mainly consists in the creation of a good target vector $\mathbf{t}$ in $\mathbb{R}^{\nu+1}$, followed by a CVP computation in the lattice L.

Algorithm 4.1 A Euclidean division over R

Input: A fractional ideal $\mathfrak{a}$, and two elements $a \in K_{\mathbb{R}}^{\times}$ and $b \in K_{\mathbb{R}}$.
Output: A pair $(u, v) \in R \times \mathfrak{a}$.

Computing a better pair (a_1, b_1)
1: Find $s \in \mathfrak{a}^{-1} \setminus \{0\}$ such that $\|s\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(\mathfrak{a}^{-1})^{1/n}$ as in Lemma 3.11.
2: Find $y \in R \setminus \{0\}$ such that $\|ya\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(a)|^{1/n}$ as in Lemma 3.11. Define $a_1 = ya$.
3: Solve CVP in R to find $x \in R$ such that $\|b/(s \cdot a_1) - x\| \leq \mu(R)$.
4: Find $k \in \mathbb{Z} \cap [-n, n]$ such that $|\mathcal{N}(b - xsa_1 + ksa_1)| \geq |\mathcal{N}(sa_1)|$ (see Lemma 4.6).
5: Define $b_1 = b + (k - x)s \cdot a_1$.

Defining the target vector and solving CVP
6: Compute $(w_{t,j})_{j \leq r_0}$ and g_t such that $\mathfrak{a}^{-1} = \prod_j \mathfrak{p}_j^{w_{t,j}} \langle g_t \rangle$. Let $\mathbf{w}_t = (w_{t,j})_{j \leq r_0}$.
7: Let $a_t = (\log \mathcal{N}|b_1/(a_1 g_t)|)/\sqrt{n}$, θ_t be the first $r_1 + r_2$ coordinates of $\overline{\mathrm{Log}}(b_1/(a_1 g_t))$ and $\mathbf{h}_t = i_{E \cap H}(\Pi_H(\mathrm{Log}(b_1/(a_1 g_t))))$.
8: Define $\mathbf{t} = (\beta a_t \| \beta \theta_t \| \beta \mathbf{h}_t \| \mathbf{w}_t \| \mathbf{0})$.
9: Solve CVP in L with target vector $\mathbf{t}$, to obtain a vector $\mathbf{s}$.

Using $\mathbf{s}$ to create a good ring element
10: Write $\mathbf{s} = (\beta a_s \| \beta \theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s)$ and let $g_s \in K^*$ be the associated element as in Lemma 4.4.
11: Define the ideal $I = \mathfrak{a} \prod_{j : w_{s,j} - w_{t,j} < 0} \mathfrak{p}_j^{w_{t,j} - w_{s,j}} \prod_{j : w'_{s,j} < 0} \mathfrak{q}_j^{-w'_{s,j}}$.
12: Find $v \in I \setminus 0$ such that $\|v\|_\infty \leq c \cdot \mathcal{N}(I)^{1/n}$ as in Lemma 3.11.
13: Define $u' = g_s \cdot g_t \cdot v$.
14: **return** $(u'y + (k - x)sy \cdot v, v)$.

Theorem 4.7 (Heuristic)**.** *Assume that $\mathfrak{a}$ satisfies $\mathrm{c}^{-n} \leq \mathcal{N}(\mathfrak{a}) \leq \mathrm{c}^n$, with c as in Lemma 3.11. Assume also that B and r are chosen so that*

$$B \geq \max\left(100 \cdot n \cdot \log[(\mu(R) + n)c^4], \log h_K \cdot (103 \cdot \frac{\delta_0}{\delta})^2\right),$$
$$\alpha_0 := \sqrt{2\pi}\left(\left(\frac{2B}{r^{0.96}}\right)^B \cdot \delta B (\det \Lambda) h_K\right)^{1/n} \leq \frac{\varepsilon}{43 \cdot \sqrt{n} \cdot (\mu(R) + n)\mathrm{c}^4 \cdot 2^{0.55 \cdot \delta \cdot B/n}},$$

for some $\varepsilon > 0$. Assume also that $\alpha_0 \leq (\ln n)/(12n^2)$, and set the scaling parameter β of $\mathbf{B}_L$ as in Heuristic 4.5. Then, under Heuristic 4.5 and the heuristics of Lemma 3.11, Algorithm 4.1 outputs a pair $(u, v) \in R \times \mathfrak{a}$ with

$$\|ua + vb\|_\infty \leq \varepsilon \cdot \|a\|_\infty,$$
$$\|v\|_\infty \leq \mathrm{c} \cdot 2^{0.55 \cdot \delta \cdot B/n}.$$

Apart from the CVP calls in R, L_K and L, Algorithm 4.1 runs in quantum polynomial time.

Proof. Throughout the proof, we keep the notations of Algorithm 4.1.

We first prove that $(u, v) \in R \times \mathfrak{a}$. As $s \in \mathfrak{a}^{-1}$ and $x, k, y \in R$, it suffices to prove that $(u', v) \in R \times \mathfrak{a}$. By definition of g_t and g_s, we have

$$\langle g_s g_t \rangle = \mathfrak{a}^{-1} \prod_j \mathfrak{p}_j^{w_{s,j} - w_{t,j}} \prod_j \mathfrak{q}_j^{w'_{s,j}} = J \cdot I^{-1},$$

with $J = \prod_{j : w_{s,j} - w_{t,j} > 0} \mathfrak{p}_j^{w_{s,j} - w_{t,j}} \prod_{j : w'_{s,j} > 0} \mathfrak{q}_j^{w'_{s,j}}$. As the $\mathfrak{p}_j$'s and $\mathfrak{q}_j$'s are integral ideals, we see that $J \subseteq R$ and $I \subseteq \mathfrak{a}$. As $v \in I$, we obtain that $v \in \mathfrak{a}$. Since $g_s \cdot g_t \in JI^{-1}$ and $v \in I$, we also have $u' = g_s g_t v \in JI^{-1}I = J \subseteq R$. This gives our first claim.

As a preliminary step towards bounding $\|ua + bv\|_\infty = \|u'a_1 + vb_1\|_\infty$, we study the sizes of a_1 and b_1. Using the equality $b_1 = b - xsa_1 + ksa_1$, we have

$$\|b_1\|_\infty \leq (\|b/(sa_1) - x\|_\infty + |k|) \cdot \|sa_1\|_\infty \leq (\mu(R) + n) \cdot \|s\|_\infty \cdot \|a_1\|_\infty.$$

By definition of a_1, we have $\|a_1\|_\infty \leq \mathrm{c}\|a\|_\infty$. By assumption on $\mathfrak{a}$, we also have $\|s\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(\mathfrak{a}^{-1})^{1/n} \leq \mathrm{c}^2$. Hence, we obtain

$$\|b_1\|_\infty \leq (\mu(R) + n)\mathrm{c}^3\|a\|_\infty.$$

Now, by definition of a_1, we know that $\|a_1\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(a_1)|^{1/n}$. Hence, we obtain

$$\mathrm{c}^{-1} \leq |\mathcal{N}(b_1/a_1)|^{1/n} \leq (\mu(R) + n) \cdot \mathrm{c}^3.$$

The left inequality is provided by the choice of k at Step 4 (and the fact that $\mathcal{N}(s) \geq \mathcal{N}(\mathfrak{a}^{-1})$).

To bound $\|u'a_1 + vb_1\|_\infty$, we estimate the closeness of $\mathbf{t}$ and $\mathbf{s}$. If $\mathbf{t}$ was in $\mathrm{Span}(L)$, then we could apply Heuristic 4.5. As this is not necessarily the case, in the proof of Lemma 4.8, we first compute the distance between $\mathbf{t}$ and $\mathrm{Span}(L)$.

Lemma 4.8 (Heuristic)**.** *Under the assumptions of Theorem 4.7, we have* $\|\mathbf{s} - \mathbf{t}\|_2 \leq \sqrt{1.06 \cdot B}$.

This lemma implies that

$$\|(a_s\|\theta_s\|\mathbf{h}_s) - (a_t\|\theta_t\|\mathbf{h}_t)\|_2 \leq \sqrt{1.06 \cdot B}/\beta \leq 15 \cdot \sqrt{n} \cdot \alpha_0.$$

By definition of $\mathbf{t}$ and construction of L, this means that

$$\|\overline{\mathrm{Log}}(g_t g_s \cdot a_1/b_1)\|_2 = \|(a_s\|\theta_s\|\mathbf{h}_s) - (a_t\|\theta_t\|\mathbf{h}_t)\|_2 \leq 15 \cdot \sqrt{n} \cdot \alpha_0.$$

Recall that $u'/v = g_t g_s$. Hence we have $\|\overline{\mathrm{Log}}(u'a_1) - \overline{\mathrm{Log}}(vb_1)\|_\infty \leq 15 \cdot \sqrt{n} \cdot \alpha_0$. Using Lemma 4.3, we deduce that

$$\begin{aligned}\|u'a_1 - vb_1\|_\infty &\leq (\mathrm{e}^{15 \cdot \sqrt{2n} \cdot \alpha_0} - 1) \cdot \|b_1\|_\infty \cdot \|v\|_\infty \\ &\leq 43 \cdot \sqrt{n} \cdot \alpha_0 \cdot \|b_1\|_\infty \cdot \|v\|_\infty,\end{aligned}$$

where we used the fact that $\alpha_0 \leq (\ln n)/(12n^2)$ and so the exponent should be smaller than $(\ln 2)/\sqrt{2}$ for n large enough. We have already bounded $\|b_1\|_\infty$. Let us now bound $\|v\|_\infty$. By definition of v, we have $\|v\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(I)^{1/n}$. The task then amounts to giving an upper bound on $\mathcal{N}(I)$. As $IJ = \mathfrak{a} \cdot \prod_j \mathfrak{p}_j^{|w_{s,j} - w_{t,j}|} \cdot \prod_j \mathfrak{q}_j^{|w'_{s,j}|}$, we have:

$$\begin{aligned}\log \mathcal{N}(IJ) &= \log \mathcal{N}(\mathfrak{a}) + \sum_j |w_{s,j} - w_{t,j}| \log \mathcal{N}(\mathfrak{p}_j) + \sum_j |w'_{s,j}| \cdot \log \mathcal{N}(\mathfrak{q}_j) \\ &\leq \log \mathcal{N}(\mathfrak{a}) + \|\mathbf{w}_s - \mathbf{w}_t\|_1 \cdot \delta_0 + \|\mathbf{w}'_s\|_1 \cdot \delta\end{aligned}$$

Recall from Lemma 4.8 that we have $\|\mathbf{s} - \mathbf{t}\|_2 \leq \sqrt{1.06 \cdot B}$. This implies that $\|\mathbf{w}_s - \mathbf{w}_t\|_2, \|\mathbf{w}'_s\|_2 \leq \sqrt{1.06 \cdot B}$. Note that

$$\|\mathbf{w}_s - \mathbf{w}_t\|_1 \leq \sqrt{r_0}\|\mathbf{w}_s - \mathbf{w}_t\|_2 \leq 1.03 \cdot \sqrt{B \cdot r_0} \leq 0.01 \cdot \frac{\delta}{\delta_0} \cdot B,$$

by assumption on B and the fact that $r_0 \leq \log h_K$. For $\mathbf{w}'_s$, we use the fact that it has integer coordinates to obtain $\|\mathbf{w}'_s\|_1 \leq \|\mathbf{w}'_s\|_2^2 \leq 1.06 \cdot B$ (see Equation 2.3). We thus obtain

$$\log \mathcal{N}(IJ) \leq \log \mathcal{N}(\mathfrak{a}) + 1.07 \cdot \delta \cdot B.$$

As J is integral, this gives an upper bound on $\mathcal{N}(I)$. However this upper bound is not sufficient for our purposes. We improve it by giving an upper bound on $\log \mathcal{N}(IJ^{-1})$, using the fact that the ideal IJ^{-1} is designed to have an algebraic norm close to the one of a_1/b_1. More precisely, it is worth recalling that $I^{-1}J = \langle g_s g_t \rangle$, and that $\|\overline{\mathrm{Log}}(g_t g_s \cdot a_1/b_1)\|_2 \leq 15 \cdot \sqrt{n} \cdot \alpha_0$. Looking at the first coordinate of

the $\overline{\mathrm{Log}}$ vector and multiplying it by $\sqrt{n}$ gives that $|\log|\mathcal{N}(g_s g_t)| + \log|\mathcal{N}(a_1/b_1)|\,| \leq 15 \cdot n \cdot \alpha_0$. This gives us

$$\log \mathcal{N}(IJ^{-1}) \leq |\log|\mathcal{N}(a_1/b_1)|| + 15 \cdot n \cdot \alpha_0$$

Combining the bounds on $\log \mathcal{N}(IJ)$ and $\log \mathcal{N}(IJ^{-1})$, we finally obtain that

$$\log \mathcal{N}(I) \leq \frac{1}{2} \cdot |\log \mathcal{N}(\mathfrak{a})| + 0.535 \cdot \delta \cdot B + \frac{1}{2} \cdot |\log|\mathcal{N}(a_1/b_1)|| + 7.5 \cdot n \cdot \alpha_0.$$

We have seen that $1 \leq |\mathcal{N}(b_1/a_1)|^{1/n} \leq (\mu(R)+n) \cdot \mathrm{c}^3$. Finally, recall that $\mathrm{c}^{-n} \leq \mathcal{N}(\mathfrak{a}) \leq \mathrm{c}^n$. Hence, we conclude that $|\log|\mathcal{N}(a_1/b_1)|| + |\log \mathcal{N}(\mathfrak{a})| \leq n \cdot \log((\mu(R)+n) \cdot \mathrm{c}^4) \leq 0.01 \cdot B$ by assumption on B. Recall that we assumed that $\alpha_0 \leq (\ln n)/(12n^2) \leq 1/n$. Hence, we have $n \cdot \alpha_0 \leq 1$. Using the fact that $B \geq 750$ (which is implied by the second term in the max), we obtain $7.5 \cdot n \cdot \alpha_0 \leq 0.01 \cdot B$. We conclude that

$$\log \mathcal{N}(I) \leq 0.55 \cdot \delta \cdot B.$$

Collecting terms and using the assumptions, this allows us to write

$$\begin{aligned} \|u' a_1 - v b_1\|_\infty &\leq 43 \cdot \sqrt{n} \cdot \alpha_0 \cdot \|b_1\|_\infty \cdot \|v\|_\infty \\ &\leq \alpha_0 \cdot 43 \cdot \sqrt{n} \cdot 2^{0.55 \cdot \delta \cdot B/n} \cdot (\mu(R)+n)\mathrm{c}^4 \|a\|_\infty \\ &\leq \varepsilon \cdot \|a\|_\infty. \end{aligned}$$

Finally, the run-time bound follows by inspection. □

Proof of Lemma 4.8. By the Pythagorean theorem, if $\mathbf{t}_L$ is the orthogonal projection of $\mathbf{t}$ onto the span of L, then we have

$$\|\mathbf{t} - \mathbf{s}\|_2^2 = \|\mathbf{t} - \mathbf{t}_L\|^2 + \|\mathbf{t}_L - \mathbf{s}\|^2 = \mathrm{dist}(\mathbf{t}, \mathrm{Span}(L))^2 + \|\mathbf{t}_L - \mathbf{s}\|^2.$$

This quantity is minimal when $\|\mathbf{t}_L - \mathbf{s}\|$ is minimal, and so the closest point to $\mathbf{t}$ in L is also the closest point to $\mathbf{t}_L$.

First, observe that $\mathrm{Span}(L)$ (of dimension ν) is exactly

$$\Big\{(\beta a \| \beta\theta \| \beta \mathbf{h} \| \mathbf{w} \| \mathbf{w}') \in \mathbb{R}^{\nu+1} : \; a = \sum_j w_j \frac{\log \mathcal{N}(\mathfrak{p}_j)}{\sqrt{n}} + \sum_j w'_j \frac{\log \mathcal{N}(\mathfrak{q}_j)}{\sqrt{n}}\Big\}.$$

Define $\mathbf{v} = (-\sqrt{n}/\beta, 0, \ldots, 0, \log \mathcal{N}(\mathfrak{p}_1), \ldots, \log \mathcal{N}(\mathfrak{q}_r))^T$, where the block of zeros has dimension $2(r_1 + r_2) - 1$. It is orthogonal to $\mathrm{Span}(L)$ and satisfies $\|\mathbf{v}\| > \sqrt{r}$, as each one of the last r coefficients is ≥ 1. Hence

$$\begin{aligned} \mathrm{dist}(\mathbf{t}, \mathrm{Span}(L)) = \frac{|\langle \mathbf{v}, \mathbf{t} \rangle|}{\|\mathbf{v}\|} &< \frac{|-\sqrt{n} a_t + \sum_j w_{t,j} \log \mathcal{N}(\mathfrak{p}_j)|}{\sqrt{r}} \\ &= \frac{|-\log|\mathcal{N}(b_1/(a_1 g_t))| - \log \mathcal{N}(\mathfrak{a}) - \log|\mathcal{N}(g_t)||}{\sqrt{r}} \\ &= \frac{|\log|\mathcal{N}(a_1/b_1)| - \log \mathcal{N}(\mathfrak{a})|}{\sqrt{r}}. \end{aligned}$$

We have seen in the proof of Theorem 4.7 that $|\log|\mathcal{N}(a_1/b_1)|| \leq n \cdot \log((\mu(R)+n) \cdot \mathrm{c}^3)$. By assumption on $\mathfrak{a}$, we have that $|\log|\mathcal{N}(a_1/b_1)|| + |\log \mathcal{N}(\mathfrak{a})| \leq n \cdot \log((\mu(R)+n) \cdot \mathrm{c}^4)$. The latter bound is $\leq 0.01 \cdot B$. So we obtain that $\mathrm{dist}(\mathbf{t}, \mathrm{Span}(L)) \leq 0.01 \cdot B/\sqrt{r} \leq \sqrt{0.01 \cdot B}$.

As $\|\mathbf{t} - \mathbf{t}_L\|_2 \leq 0.01 \cdot B/\sqrt{r}$ and the last r entries of $\mathbf{t}$ are zero, we have that the last r entries of $\mathbf{t}_L$ have euclidean norm $\leq 0.01 \cdot B/\sqrt{r}$. We can hence apply Heuristic 4.5 to $\mathbf{t}_L$, which gives us $\|\mathbf{t} - \mathbf{s}\|_2^2 = \|\mathbf{t} - \mathbf{t}_L\|^2 + \|\mathbf{t}_L - \mathbf{s}\|^2 \leq 0.01 \cdot B + 1.05 \cdot B$. □

We observe that the parameters r and B of Theorem 4.7 can be instantiated as $B = \widetilde{O}(\log|\Delta| + n \log \mu(R))$ and $r^{0.96} = \Theta((1/\varepsilon)^{n/B} \cdot B \cdot 2^{0.55\delta})$. Thanks to the 0.55 in the exponent, this choice of r is compatible with the condition $r \leq 2^\delta/(\log|\Delta|)^{\eta'}$ for some $\eta' > 0$ which was required for the construction of the lattice L (recall that we want r prime ideals of norm smaller than 2^δ). We note also that the

constants 0.96 and 0.55 appearing in the exponent can be chosen as close as we want to 1 and 0.5 respectively, by adapting the argument above. Hence, assuming $(1/\varepsilon)^{n/B} = O(1)$, we expect to be able to choose 2^δ as small as $B^{2+\eta}$ for any $\eta > 0$. Overall, the following corollary gives an instantiation of Theorem 4.7 with parameters that are relevant to our upcoming divide-and-swap algorithm.

Corollary 4.9 (Heuristic)**.** *Let $\varepsilon = 1/2^{\widetilde{O}(\log|\Delta|)/n}$. For any $\eta > 0$, there exists a lattice L' of dimension $\widetilde{O}((\log|\Delta| + n\log\mu(R))^{2+\eta})$, an upper bound $C = 2^{\widetilde{O}(\log|\Delta| + n\log\mu(R))/n}$ and an algorithm $\mathcal{A}$ that achieve the following. Under Heuristic 4.5 and the heuristics of Lemma 3.11, algorithm $\mathcal{A}$ takes as inputs $a \in K_{\mathbb{R}}^\times$, $b \in K_{\mathbb{R}}$ and an ideal $\mathfrak{a}$ satisfying $\mathrm{c}^{-n} \leq \mathcal{N}(\mathfrak{a}) \leq \mathrm{c}^n$, and outputs $u, v \in R \times \mathfrak{a}$ such that*

$$\|ua + bv\|_\infty \leq \varepsilon \cdot \|a\|_\infty$$
$$\|v\|_\infty \leq C.$$

If given access to an oracle solving the closest vector problem in L' in polynomial time, and when restricted to inputs a, b belonging to K, Algorithm 4.1 runs in quantum polynomial time.

Proof. Consider an instantiation of Theorem 4.7 with $B = \widetilde{\Theta}(\log|\Delta| + n\log\mu(R))$, $\delta = 2.451 \cdot \log_2 B$ and $r = B^{2.45}$. This choice of parameters asymptotically satisfies $(\log|\Delta|)^2 \leq r \leq 2^\delta/(\log|\Delta|)^{\eta'}$ for some $\eta' > 0$, which was required for the generation of the lattice L. It also satisfies the constraints of Theorem 4.7. It can be checked by proof inspection that the constant 2.45 can be adapted to $2 + \eta$ for an arbitrary $\eta > 0$, which allows to obtain the asymptotic parametrization of the statement. Finally, observe that the algorithm relies on oracles that solve CVP in R, L_K and L. We can consider the direct sum of these lattices, to obtain a single lattice L'. □

4.3.5 The divide-and-swap algorithm

In this section, we describe a divide-and-swap algorithm, which takes as input a pseudo-basis of a rank-2 module and outputs a short non-zero vector of this module (for the algebraic norm). In order to do so, we will need to link the Euclidean and algebraic norms of vectors appearing during the execution, and limit the degree of freedom of the ideal coefficients. For this purpose we introduce the notion of scaled pseudo-bases.

Definition 4.10. A pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq m}$, with $I_i \subset K$ and $\mathbf{b}_i \in K_{\mathbb{R}}^s$ for all $i \leq m$, is said strongly scaled if, for all $i \leq m$,

$$R \subseteq I_i, \quad \mathcal{N}(I_i) \geq \mathrm{c}^{-n} \text{ and } \|r_{ii}\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(r_{ii} I_i)^{1/n},$$

where c is as in Lemma 3.11 and r_{ii} refers to the QR-factorization of the matrix formed by the $\mathbf{b}_i$ vectors (see Section 2.3.7.1).

We now describe an algorithm which takes as input a pseudo-basis of a module M and outputs a strongly scaled pseudo-basis of the same module. We state it in dimension m but will only use it in dimension 2 in this chapter (in the next section, we use a very similar algorithm in dimension m).

Algorithm 4.2 Strongly scaling the ideals.

Input: A pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq m}$ of a module M.
Output: A strongly scaled pseudo-basis $((I'_i, \mathbf{b}'_i))_{i \leq m}$ of M.

1: **for** $i = 1$ **to** m **do**
2: Use Lemma 3.11 to find $s_i \in r_{ii} \cdot I_i \setminus \{0\}$ such that $\|s_i\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(r_{ii} I_i)^{1/n}$;
3: Write $s_i = r_{ii} \cdot x_i$, with $x_i \in I_i$;
4: Define $I'_i = I_i \cdot \langle x_i \rangle^{-1}$ and $\mathbf{b}'_i = x_i \mathbf{b}_i$.
5: **end for**
6: **return** $((I'_i, \mathbf{b}'_i))_{i \leq m}$.

Lemma 4.11. *Algorithm 4.2 outputs a strongly scaled pseudo-basis of the module M generated by the input pseudo-basis and preserves the $\mathcal{N}(r_{ii} I_i)$'s. If given access to an oracle that solves CVP in the lattice L_K of Lemma 3.11, and if $M \subseteq R^s$, then it runs quantumly in time polynomial in the input bit-length and in* $\log|\Delta|$.

Proof. The algorithm scales each column of the pseudo-matrix by some factor $x_i \in K$ and scales the corresponding ideal accordingly. This operation preserves the spanned module. Further, the fact that x_i belongs to K implies that the ideal I_i' remains a fractional ideal of K.

Fix $i \leq m$. By choice of s_i, we know that $\mathcal{N}(s_i) \leq \mathrm{c}^n \cdot \mathcal{N}(r_{ii} I_i)$. We hence have $\mathcal{N}(x_i) \leq \mathrm{c}^n \cdot \mathcal{N}(I_i)$ and $\mathcal{N}(I_i') = \mathcal{N}(I_i)/\mathcal{N}(x_i) \geq \mathrm{c}^{-n}$. Further, as $x_i \in I_i$, we have $R \subseteq I_i'$, which implies that $\mathcal{N}(I_i') \leq 1$. This proves the conditions on I_i'. Moreover, we have $r_{ii}' = x_i r_{ii} = s_i$, hence $\|r_{ii}'\| = \|s_i\| \leq \mathrm{c} \cdot \mathcal{N}(r_{ii} I_i)^{1/n} = \mathrm{c} \cdot \mathcal{N}(r_{ii}' I_i')^{1/n}$. This proves the bound on $\|r_{ii}'\|$.

Now, observe that $\mathbf{b}_i'$ is $K_{\mathbb{R}}$-colinear to $\mathbf{b}_i$. Hence, replacing a vector $\mathbf{b}_i$ by $\mathbf{b}_i'$ does not impact r_{jj} for $j \neq i$. The quantities $\mathcal{N}(r_{jj} I_j)$ are therefore preserved through the execution of the algorithm.

The bound on the running time follows by inspection of the algorithm. □

We can now describe Algorithm 4.3, our divide-and-swap algorithm. During the execution of the algorithm, the R-factor of the current matrix $(\mathbf{b}_1 | \mathbf{b}_2)$ is always computed. The algorithm is very similar to the LLL algorithm in dimension 2, except for Step 4, which is specific to this algorithm. This step ensures that when we swap the vectors, we still obtain a pseudo-basis of the input module. This seems necessary, as our Euclidean division over R involves a multiplication of the second vector by a ring element, and hence the new vector and the second pseudo-basis vector may not span the whole module anymore. At Step 4, note that the gcd is well-defined, as $\langle u \rangle$ and $\langle v \rangle \mathfrak{a}^{-1}$ are integral ideals. As an alternative to Step 4, we could use Lemma 2.10.

Algorithm 4.3 Divide-and-swap.

Input: A pseudo-basis $\begin{bmatrix} \mathfrak{a}_1 & \mathfrak{a}_2 \\ \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}$ of a module $M \subset K_{\mathbb{R}}^2$.
Output: A vector $\mathbf{v} \in M$.

1: **while** $(\gamma/\mathrm{c})^n \mathcal{N}(r_{22}\mathfrak{a}_2) < \mathcal{N}(r_{11}\mathfrak{a}_1)$ **do**
2: Strongly scale the pseudo-basis $\begin{bmatrix} \mathfrak{a}_1 & \mathfrak{a}_2 \\ \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}$ using Algorithm 4.2.
3: Apply Algorithm 4.1 to $(a, b, \mathfrak{a}) = (r_{11}, r_{12}, \mathfrak{a}_2 \cdot \mathfrak{a}_1^{-1})$ and $\varepsilon = 1/(4\mathrm{c}\sqrt{n})$. Let (u, v) be the output.
4: Let $\mathfrak{b} = \gcd(\langle u \rangle, \langle v \rangle \mathfrak{a}^{-1})$, find $x \in \mathfrak{a}^{-1}\mathfrak{b}^{-1}$ and $y \in \mathfrak{b}^{-1}$ such that $uy - vx = 1$.
5: Update $(\mathbf{b}_1, \mathbf{b}_2) \leftarrow (u\mathbf{b}_1 + v\mathbf{b}_2, x\mathbf{b}_1 + y\mathbf{b}_2)$ and $(\mathfrak{a}_1, \mathfrak{a}_2) \leftarrow (\mathfrak{a}_1\mathfrak{b}^{-1}, \mathfrak{a}_2\mathfrak{b})$.
6: **end while**
7: Strongly scale the pseudo-basis $\begin{bmatrix} \mathfrak{a}_1 & \mathfrak{a}_2 \\ \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}$ using Algorithm 4.2.
8: **return** $\mathbf{b}_1$

Lemma 4.12. *Let $\gamma \geq 4 \cdot C \cdot \sqrt{n} \cdot \mathrm{c}^2$, where C is as in Corollary 4.9. Then, given as input a pseudo-basis of a rank-2 module $M \subset K_{\mathbb{R}}^2$, Algorithm 4.3 outputs a vector $\mathbf{v} \in M \setminus \{\mathbf{0}\}$ such that $\mathcal{N}(\mathbf{v}) \leq \gamma^n \lambda_1^{\mathcal{N}}(M)$. Further, if $M \subseteq R^s$ and Algorithms 4.1 and 4.2 run in polynomial time, then Algorithm 4.3 runs in time polynomial in the input bit-length and in* $\log |\Delta|$.

Proof. Let us first prove that the pseudo-basis $\begin{bmatrix} \mathfrak{a}_1 & \mathfrak{a}_2 \\ \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}$ we have throughout the execution of the algorithm remains a pseudo-basis of M. By Lemma 4.11, this property is preserved through Steps 2 and 7. It remains to prove it for Step 5. At this step, we multiply $\begin{pmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{pmatrix}$ on the right by $\mathbf{U} := \begin{pmatrix} u & x \\ v & y \end{pmatrix}$. Let $\mathfrak{a}_1, \mathfrak{a}_2$ (resp. $\mathfrak{a}_1' = \mathfrak{a}_1\mathfrak{b}^{-1}, \mathfrak{a}_2' = \mathfrak{a}_2\mathfrak{b}$) denote the coefficient ideals at the start (resp. completion) of Step 5. We know from the preliminaries that this transformation outputs a pseudo-basis of the same module if $\mathbf{U}$ is invertible over K and $u_{ij} \in \mathfrak{a}_i(\mathfrak{a}_j')^{-1}$ and $u_{ij}' \in \mathfrak{a}_i'(\mathfrak{a}_j)^{-1}$ for all $i, j \in \{1, 2\}$, with $\mathbf{U}' = \mathbf{U}^{-1}$. In our case, because we asked that $uy - vx = 1$, then $\mathbf{U}$ is indeed invertible and we have $\mathbf{U}^{-1} = \begin{pmatrix} y & -x \\ -v & u \end{pmatrix}$. Observe that by definition of $\mathfrak{b} = \gcd(\langle u \rangle, \langle v \rangle \mathfrak{a}^{-1})$, we have $u \in \mathfrak{b}$ and $v \in \mathfrak{a} \cdot \mathfrak{b} = \mathfrak{a}_1^{-1}\mathfrak{a}_2\mathfrak{b}$. Using these properties and the fact that $x \in \mathfrak{a}_1\mathfrak{a}_2^{-1}\mathfrak{b}^{-1}$ and $y \in \mathfrak{b}^{-1}$ by definition, one can then check that all the conditions $u_{ij} \in \mathfrak{a}_i(\mathfrak{a}_j')^{-1}$ and $u_{ij}' \in \mathfrak{a}_i'(\mathfrak{a}_j)^{-1}$ are indeed satisfied.

We now prove that the output vector $\mathbf{v}$ belongs to M. As $\mathbf{v} = \mathbf{b}_1$, it suffices that $R \subseteq \mathfrak{a}_1$. This is provided by the application of Algorithm 4.2 at Step 7.

Assume now that the algorithm terminates, and let us show that the output vector $\mathbf{v}$ satisfies $\mathcal{N}(\mathbf{v}) \leq \gamma^n \lambda_1^{\mathcal{N}}(M)$. Because of the application of Algorithm 4.2 at Step 7, we have $\mathcal{N}(\mathfrak{a}_1) \geq \mathrm{c}^{-n}$. This, and the inequality $\gamma \geq \mathrm{c}$, imply that $\mathcal{N}(\mathbf{b}_1) = \mathcal{N}(r_{11}) \leq \mathrm{c}^n \cdot \mathcal{N}(r_{11}\mathfrak{a}_1) \leq \gamma^n \cdot \mathcal{N}(r_{11}\mathfrak{a}_1)$. On the other hand, because we exited the while loop, we have $(\gamma/\mathrm{c})^n \mathcal{N}(r_{22}\mathfrak{a}_2) \geq \mathcal{N}(r_{11}\mathfrak{a}_1)$ (by Lemma 4.11, Step 7 does not change the values of $\mathcal{N}(r_{11}\mathfrak{a}_1)$ and $\mathcal{N}(r_{22}\mathfrak{a}_2)$). We conclude that (using Lemma 2.9):

$$\mathcal{N}(\mathbf{b}_1) \leq \gamma^n \cdot \min(\mathcal{N}(r_{11}\mathfrak{a}_1), \mathcal{N}(r_{22}\mathfrak{a}_2)) \leq \gamma^n \lambda_1^{\mathcal{N}}(M).$$

It remains to show that the algorithm is polynomial time (assuming that Algorithms 4.1 and 4.2 are polynomial time). For this, we first prove that the number of loop iterations is polynomial. We do so by proving that $\mathcal{N}(r_{11}\mathfrak{a}_1)$ decreases by a factor $\geq 2^n$ at each iteration. As the product $\mathcal{N}(r_{11}\mathfrak{a}_1)\mathcal{N}(r_{22}\mathfrak{a}_2) = \det(M)/|\Delta|$ is constant and we stop whenever $\mathcal{N}(r_{11}\mathfrak{a}_1)$ becomes smaller than $(\gamma/\mathrm{c})^n \mathcal{N}(r_{22}\mathfrak{a}_2)$, the number of iterations is bounded by $\log \mathcal{N}(r_{11}\mathfrak{a}_1)/n$ (for the r_{11} and $\mathfrak{a}_1$ of the input).

Recall that at the end of Step 2, we have $\|r_{ii}\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(r_{ii}\mathfrak{a}_i)^{1/n}$ for $i = 1, 2$. Recall also that Algorithm 4.1 outputs u, v such that $\|ur_{11} + vr_{12}\|_\infty \leq \varepsilon \|r_{11}\|_\infty$ and $\|v\|_\infty \leq C$. The new vector $\mathbf{b}_1$ at the end of the loop iteration is $u\mathbf{b}_1 + v\mathbf{b}_2$. We compute an upper bound on its algebraic norm:

$$\begin{aligned}
\mathcal{N}(u\mathbf{b}_1 + v\mathbf{b}_2) \leq \|u\mathbf{b}_1 + v\mathbf{b}_2\|^n &= \left\| \begin{pmatrix} ur_{11} + vr_{12} \\ vr_{22} \end{pmatrix} \right\|^n \\
&\leq \left(\|ur_{11} + vr_{12}\| + \|vr_{22}\|\right)^n \\
&\leq (\sqrt{n})^n \cdot \left(\|ur_{11} + vr_{12}\|_\infty + \|vr_{22}\|_\infty\right)^n \\
&\leq (\sqrt{n})^n \cdot \left(\varepsilon \|r_{11}\|_\infty + \|v\|_\infty \cdot \|r_{22}\|_\infty\right)^n .
\end{aligned}$$

Using the facts that the basis is strongly scaled and that the condition of Step 1 is satisfied, we have:

$$\begin{aligned}
\mathcal{N}(u\mathbf{b}_1 + v\mathbf{b}_2) &\leq \mathrm{c}^n \cdot (\sqrt{n})^n \cdot \left(\varepsilon \mathcal{N}(r_{11}\mathfrak{a}_1)^{1/n} + C \cdot \mathcal{N}(r_{22}\mathfrak{a}_2)^{1/n}\right)^n \\
&\leq \mathrm{c}^n \cdot (\sqrt{n})^n \cdot (\varepsilon + C \cdot (\mathrm{c}/\gamma))^n \cdot \mathcal{N}(r_{11}\mathfrak{a}_1).
\end{aligned}$$

Now, by choice of ε and γ:

$$\mathcal{N}(u\mathbf{b}_1 + v\mathbf{b}_2) \leq \mathrm{c}^n \cdot \left(\frac{1}{4\mathrm{c}} + \frac{1}{4\mathrm{c}}\right)^n \cdot \mathcal{N}(r_{11}\mathfrak{a}_1) = 2^{-n} \cdot \mathcal{N}(r_{11}\mathfrak{a}_1).$$

Recall that $\mathfrak{a}_1$ is also updated as $\mathfrak{a}_1\mathfrak{b}^{-1}$. Hence, to conclude, we argue that $\mathcal{N}(\mathfrak{a}_1\mathfrak{b}^{-1}) \leq 1$. Note that $\mathcal{N}(\mathfrak{a}_1) \leq 1$ holds due to scaling, and that $\mathcal{N}(\mathfrak{b}) \geq 1$ holds because $\mathfrak{b}$ is integral. Overall, we obtain that $\mathcal{N}(r_{11}\mathfrak{a}_1)$ decreases by a factor $\geq 2^n$ during a loop iteration.

To complete the cost analysis, we observe that all the steps run in polynomial time, except maybe Step 4. In this step, it is a priori not obvious that the elements x and y satisfying the stated conditions even exist. The conditions can be re-stated as $uy \in \langle u \rangle \mathfrak{b}^{-1}$, $vx \in \langle v \rangle \mathfrak{a}^{-1}\mathfrak{b}^{-1}$ and $uy - vx = 1$. Hence such x, y exist if the ideals $\langle u \rangle \mathfrak{b}^{-1}$ and $\langle v \rangle \mathfrak{a}^{-1}\mathfrak{b}^{-1}$ are coprime. This is indeed the case, by construction of $\mathfrak{b}$. Further, we can compute x, y in polynomial time by computing a basis of the lattice spanned by the two ideals (which is R, as they are coprime). Finally, note that by ideal scaling and the fact that the quantity $\mathcal{N}(r_{11}\mathfrak{a}_1)$ decreases throughout the algorithm, all pseudo-bases occurring through the execution have bit-sizes polynomial in the input bit-size. □

Instantiating this lemma with the value of C obtained in Corollary 4.9, we obtain the following corollary.

Corollary 4.13 (Heuristic)**.** *For any number field K and any $\eta > 0$, there exists a lattice L' of dimension $\widetilde{O}((\log|\Delta| + n\log\mu(R))^{2+\eta})$, a choice of $\gamma = 2^{\widetilde{O}(\log|\Delta| + n\log\mu(R))/n}$ and an algorithm $\mathcal{A}$ such that the following holds. Under Heuristic 4.5 and the heuristics of Lemma 3.11, algorithm $\mathcal{A}$ takes as input a pseudo-basis of a rank-2 module $M \subset K_{\mathbb{R}}^2$, and outputs a vector $\mathbf{v} \in M$ such that $\mathcal{N}(\mathbf{v}) \leq \gamma^n \lambda_1^{\mathcal{N}}(M)$. If given access to an oracle solving the closest vector problem in L' in polynomial time, and when restricted to modules contained in K^2, Algorithm $\mathcal{A}$ runs in quantum polynomial time.*

Proof. The corollary is obtained by combining Lemma 4.12 with Corollary 4.9. To apply Corollary 4.9, we need $1/\varepsilon = 2^{\widetilde{O}(\log|\Delta|)/n}$, which is indeed the case in Algorithm 4.3. Note that the choice of ε in Algorithm 4.3 only depends on K. □

4.4 LLL-reduction of module pseudo-bases

We now proceed to explain how the divide-and-swap algorithm described above can be used to extend the LLL algorithm to any module of dimension m over some ring R. LLL-reduction of lattice bases is defined in terms of Gram-Schmidt orthogonalization (or, equivalently, QR-factorization). A basis is said LLL-reduced if two conditions are satisfied. The first one, often referred to as size-reduction condition, states that any off-diagonal coefficients r_{ij} of the R-factor should have a small magnitude compared to the diagonal coefficient r_{ii} on the same row. The second one, often referred to as Lovász' condition, states that the 2-dimensional vector $(r_{i,i}, 0)^T$ is no more than $1/\delta$ times longer than $(r_{i,i+1}, r_{i+1,i+1})^T$, for some parameter $\delta < 1$. The size-reduction condition allows to ensure that the norms of the vectors during the LLL execution and at its completion stay bounded. More importantly, in combination with Lovász' condition, it makes it impossible for $r_{i+1,i+1}/r_{i,i}$ to be arbitrarily small (for an LLL-reduced basis). The latter is the crux of both the LLL output quality and its fast termination.

4.4.1 An LLL algorithm for module lattices

When extending to rings, the purpose of the size-reduction condition is better expressed in terms of the Euclidean norm $\|\cdot\|$, whereas the bounded decrease of the r_{ii}'s is better quantified in terms of the algebraic norm $\mathcal{N}(\cdot)$. This discrepancy makes the definition of a LLL-reduction algorithm for modules difficult. In this section, we circumvent this difficulty by directly focusing on the decrease of the r_{ii}'s, deferring to later the bounding of bit-sizes.

Definition 4.14 (LLL-reducedness of a pseudo-basis)**.** A module pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq m}$ is called LLL-reduced with respect to a parameter α_K if, for all $i < m$, we have:

$$\mathcal{N}(r_{i+1,i+1}I_{i+1}) \geq \frac{1}{\alpha_K} \cdot \mathcal{N}(r_{i,i}I_i), \tag{4.1}$$

where $\mathbf{R} = (r_{i,j})_{i,j}$ refers to the R-factor of the matrix basis $\mathbf{B}$.

We first explain that LLL-reduced pseudo-bases are of interest, and we will later discuss their computation (for some value of α_K).

Lemma 4.15. *Assume that* $\begin{bmatrix} I_1 & \dots & I_m \\ \mathbf{b}_1 & \dots & \mathbf{b}_m \end{bmatrix}$ *is an LLL-reduced pseudo-basis of a module M. Then:*

$$\begin{aligned} \mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) &\leq \alpha_K^{(m-1)/2} \cdot (\mathcal{N}(\det_{K_\mathbb{R}} M))^{1/m}, \\ \mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) &\leq \alpha_K^{m-1} \cdot \lambda_1^{\mathcal{N}}(M). \end{aligned}$$

Proof. From (4.1), we obtain $\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq \alpha_K^i \mathcal{N}(r_{i,i}I_i)$ for all $i \leq m$. Taking the product over all i's gives $(\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1))^m \leq \alpha_K^{m(m-1)/2}\mathcal{N}(\det_{K_\mathbb{R}} M)$, by Lemma 2.9. The proof of the first inequality can be completed by taking the m-th root. The second inequality can be obtained by combining the last claim of Lemma 2.9 with (4.1).

□

Our LLL algorithm for modules is very similar to the one over the integers.

The algorithm proceeds by finding an approximation to a shortest non-zero element in a rank-2 module, with respect to the algebraic norm. Using Lemma 2.8, we obtain a sufficient condition on α_K such that Algorithm 4.4 terminates. In particular, if α_K is sufficiently large, then $\mathcal{N}(r_{i+1,i+1}I_{i+1}) < \frac{1}{\alpha_K}\mathcal{N}(r_{i,i}I_i)$ implies that there is a vector $\mathbf{s}$ in the local projected rank-2 module of norm significantly less than $\mathcal{N}(r_{i,i}I_i)$.

Lemma 4.16. *Take the notations of Algorithm 4.4, and consider an index $i < m$ such that $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}) < \mathcal{N}(r_{i,i}I_i)$. We have $\mathcal{N}(\mathbf{s}_i) \leq \gamma^n \sqrt{\frac{2^n\Delta_K}{\alpha_K}}\mathcal{N}(r_{i,i}I_i)$.*

Algorithm 4.4 LLL-reduction over K

Input: A pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq m}$ of a module $M \subset K^s$.
Output: An LLL-reduced pseudo-basis of M.

1: **while** there exists $i < m$ such that $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}) < \mathcal{N}(r_{i,i}I_i)$ **do**
2: Define M_i as the rank-2 module spanned by $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$, with $\mathbf{a}_i = (r_{ii}, 0)^T$ and $\mathbf{a}_{i+1} = (r_{i,i+1}, r_{i+1,i+1})^T$;
3: Find $\mathbf{s}_i \in M_i \setminus \{\mathbf{0}\}$ such that $\mathcal{N}(\mathbf{s}_i) \leq \gamma^n \cdot \lambda_1^{\mathcal{N}}(M_i)$;
4: Set $\mathbf{s}_{i+1} = \mathbf{a}_i$ if it is linearly independent with $\mathbf{s}_i$, and $\mathbf{s}_{i+1} = \mathbf{a}_{i+1}$ otherwise;
5: Call the algorithm of Lemma 2.10 with $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ and $(\mathbf{s}_i, \mathbf{s}_{i+1})$ as inputs, and let $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ denote the output;
6: Update $I_i := I'_i$, $I_{i+1} := I'_{i+1}$ and $[\mathbf{b}_i|\mathbf{b}_{i+1}] := [\mathbf{b}_i|\mathbf{b}_{i+1}] \cdot \mathbf{A}^{-1} \cdot \mathbf{A}'$ (where $\mathbf{A} = [\mathbf{a}_i|\mathbf{a}_{i+1}]$ and $\mathbf{A}' = [\mathbf{a}'_i|\mathbf{a}'_{i+1}]$).
7: **end while**
8: **return** $((I_i, \mathbf{b}_i))_{i\leq m}$.

Proof. Using Lemma 2.8 applied to the rank-2 module M_i, we have

$$\mathcal{N}(\mathbf{s}_i) \leq \gamma^n 2^{n/2} \Delta_K^{1/2} \big(\mathcal{N}(r_{i,i}I_i)\mathcal{N}(r_{i+1,i+1}I_{i+1})\big)^{1/2}.$$

Using the assumption allows to complete the proof.

□

We are now ready to prove the main result of this section.

Theorem 4.17. *Assume that Step 3 of Algorithm 4.4 is implemented with some algorithm $\mathcal{O}$ for some parameter γ. Assume that $\alpha_K > \gamma^{2n} 2^n \Delta_K$. Then Algorithm 4.4 terminates and outputs an LLL-reduced pseudo-basis of M. Further, the number of loop iterations is bounded by*

$$\frac{m(m+1)}{\log(\alpha_K/(\gamma^{2n}2^n\Delta_K))} \cdot \log \frac{\max \mathcal{N}(r_{ii}I_i)}{\min \mathcal{N}(r_{ii}I_i)},$$

where the I_i's and r_{ii}'s are those of the input pseudo-basis.

Proof. We first show that at every stage of the algorithm, the current pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq m}$ is a pseudo-basis of M. For this, it suffices to show that the operations performed on it at Step 6 preserves this property. For Step 6, it is provided by the fact that $\mathbf{A}^{-1} \cdot \mathbf{A}'$ maps the pseudo-basis $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ into the pseudo-basis $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ of the same rank-2 module (by Lemma 2.10). Applying the same transformation to $((I_i, \mathbf{b}_i), (I_{i+1}, \mathbf{b}_{i+1}))$ preserves the spanned rank-2 module. The correctness of Algorithm 4.4 is implied by termination and the above.

We now prove a bound on the number of loop iterations, which will in particular imply termination. Consider the quantity

$$\Pi := \prod_{i\leq m} \mathcal{N}(r_{ii}I_i)^{m-i+1}.$$

This quantity if bounded from above by $\max \mathcal{N}(r_{ii}I_i)^{m(m+1)/2}$ and from below by $\min \mathcal{N}(r_{ii}I_i)^{m(m+1)/2}$. Below, we show that Π never increases during the execution of the algorithm, and that at every iteration of the while loop, it decreases by a factor $> \sqrt{\alpha_K/(\gamma^{2n}2^n\Delta_K)}$. We also show that the value $\min \mathcal{N}(r_{ii}I_i)$ can only increase during the execution of the algorithm, hence the lower bound above holds with respect to the input r_{ii} and I_i at every step of the algorithm. Combining the decrease rate with the above upper and lower bounds, this implies that the number of loop iterations is bounded by

$$\frac{m(m+1)}{\log(\alpha_K/(\gamma^{2n}2^n\Delta_K))} \cdot \log \frac{\max \mathcal{N}(r_{ii}I_i)}{\min \mathcal{N}(r_{ii}I_i)},$$

where the I_i's and r_{ii}'s are those of the input pseudo-basis.

Consider an iteration of the while loop, working at index i. We have $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}) < \mathcal{N}(r_{i,i}I_i)$. Step 6 is the only one that may change Π. Observe that we have

$$\Pi = \prod_{j\leq m} \mathcal{N}\left(\det_{K_{\mathbb{R}}}\left(((I_i, \mathbf{b}_i))_{i\leq j}\right)\right).$$

During the loop iteration, none of the m modules in the expression above changes, except possibly the i-th one. Now, note that

$$\mathcal{N}\left(\det_{K_{\mathbb{R}}}\left(((I_k, \mathbf{b}_k))_{k\le i}\right)\right) = \prod_{k\le i}\mathcal{N}(r_{kk}I_k).$$

During the loop iteration under scope, only the i-th term in this product may change. At Step 6, it is updated from $\mathcal{N}(r_{ii}I_i)$ to $\mathcal{N}(I_i')\mathcal{N}(\mathbf{a}_i')$. By Lemma 2.10, we have $\mathcal{N}(I_i') \le 1$ and $\mathbf{a}_i' = \mathbf{s}_i$. Now, by Lemma 4.16, we have that $\mathcal{N}(\mathbf{s}_i) \le \gamma^n \sqrt{\frac{2^n\Delta_K}{\alpha_K}}\mathcal{N}(r_{ii}I_i)$. Overall, this gives that $\mathcal{N}(r_{ii}I_i)$ and hence Π decrease by a factor $> \sqrt{\alpha_K/(\gamma^{2n}2^n\Delta_K)}$.

□

4.4.2 Handling bit-sizes

In terms of bit-sizes of the diverse quantities manipulated during the execution of the algorithm, there can be several sources of bit-size growth. Like in the classical LLL-algorithm, the Euclidean norms of off-diagonal coefficients r_{ij} for $i < j$ could grow during the execution. We handle this using a generalized size-reduction algorithm. Other annoyances are specific to the number field setup. There is too much freedom in representing a rank-1 module $I\mathbf{v}$: scaling the ideal I by some $x \in K$ and dividing $\mathbf{v}$ by the same x preserves the module. In the extreme case, it could cost an arbitrarily large amount of space, even to store a trivial rank-1 module such as $R \cdot (1, 0, \ldots, 0)^T$, if such a bad scaling is used (e.g., using such an x with large algebraic norm). Finally, even if the ideal I is "scaled", we can still multiply $\mathbf{v}$ by a unit: this preserves the rank-1 module, but makes its representation longer.[2] We introduce the notion of scaled pseudo-bases, which is very similar to (but weaker than) the notion of strongly scaled pseudo-basis we defined in Section 4.3.5.

Definition 4.18. A pseudo-basis $((I_i, \mathbf{b}_i))_{i\le m}$, with $I_i \subset K$ and $\mathbf{b}_i \in K_{\mathbb{R}}^s$ for all $i \le m$, is said scaled if, for all $i \le m$,

$$R \subseteq I_i, \quad \mathcal{N}(I_i) \ge 2^{-n^2}\Delta_K^{-1/2} \text{ and } \quad \|r_{ii}\| \le 2^n\Delta_K^{1/(2n)}\mathcal{N}(r_{ii}I_i)^{1/n}.$$

It is said size-reduced if $\|r_{ij}/r_{ii}\| \le (4n)^n\Delta_K^{1/2}$ for all $i < j \le m$.

Note that if $((I_i, \mathbf{b}_i))_{i\le m}$ is scaled, then $\mathcal{N}(I_i) \le 1$ for all $i \le m$. Further, if the spanned module is contained in R^s, then $\mathbf{b}_i \in R^s$ for all $i \le m$. Algorithm 4.5 transforms any pseudo-basis into a scaled pseudo-basis of the same module. It is a direct adaptation of Algorithm 4.2 in which the algorithm from Lemma 3.11 is replaced by the LLL algorithm. Compared to Algorithm 4.2, this algorithm is not heuristic and does not require an oracle solving CVP in a fixed lattice. It outputs a scaled pseudo-basis, which is weaker than a strongly scaled pseudo-basis, but will be sufficient for our purposes in this section.

Algorithm 4.5 Scaling the ideals.

Input: A pseudo-basis $((I_i, \mathbf{b}_i))_{i\le m}$ of a module M.
Output: A scaled pseudo-basis $((I_i', \mathbf{b}_i'))_{i\le m}$ of M.

1: **for** $i = 1$ **to** m **do**
2: Use LLL to find $s_i \in r_{ii} \cdot I_i \setminus \{0\}$ such that $\|s_i\| \le 2^n\Delta_K^{1/(2n)}\mathcal{N}(r_{ii}I_i)^{1/n}$;
3: Write $s_i = r_{ii} \cdot x_i$, with $x_i \in I_i$;
4: Define $I_i' = I_i \cdot \langle x_i\rangle^{-1}$ and $\mathbf{b}_i' = x_i\mathbf{b}_i$.
5: **end for**
6: **return** $((I_i', \mathbf{b}_i'))_{i\le m}$.

Lemma 4.19. *Algorithm 4.5 outputs a scaled pseudo-basis of the module M generated by the input pseudo-basis and preserves the $\mathcal{N}(r_{ii}I_i)$'s. If $M \subseteq R^s$, then it runs in time polynomial in the input bit-length and in $\log\Delta_K$.*

[2] Note that ideal scaling and size-reduction have been suggested in [FS10, Se. 4.1], but without a complexity analysis (polynomial complexity was claimed but not proved).

Proof. The proof is a direct adaptation of the proof of Lemma 4.11, replacing c by $2^n\Delta_K^{1/(2n)}$. Observe that the determinant of the canonical embedding of $r_{ii}I_i$ is $\Delta_K^{1/2}\mathcal{N}(r_{ii}I_i)$ and its dimension is n, so LLL can indeed be used to find $s_i \in r_{ii}\cdot I_i \setminus \{0\}$ such that $\|s_i\| \leq 2^n\Delta_K^{1/(2n)}\mathcal{N}(r_{ii}I_i)^{1/n}$.

If $M \subseteq R^s$, by Lemmas 2.23, 2.24 and 2.25, all the operations performed in the algorithm can be done in polynomial time. Hence the whole algorithm runs in polynomial time. □

Algorithm 4.6 aims at size-reducing a scaled pseudo-basis. It relies on a $\lfloor\cdot\rceil_R$ operator which takes as input a $y \in K_{\mathbb{R}}$ and rounds it to some $k \in R$ by writing $y = \sum y_i r_i$ for some y_i's in $\mathbb{R}$, and rounding each y_i to the nearest integer: $k = \sum k_i r_i = \sum \lfloor y_i \rceil r_i$ (remember that the r_i's form an LLL-reduced basis of R). For computations, we will apply this operator numerically, so that we may not have $\max_i |k_i - y_i| \leq 1/2$ but, with a bounded precision computation, we can ensure that $\max_i |k_i - y_i| \leq 1$

Algorithm 4.6 Size-reduction.

Input: A scaled pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq m}$ of a module M.
Output: A size-reduced pseudo-basis of M.

```
1: for j = 1 to m do
2:   for i = j − 1 to 1 do
3:     Compute x_i = ⌊r_ij/r_ii⌉_R;
4:     b_j := b_j − x_i b_i.
5:   end for
6: end for
7: return ((I_i, b_i))_{i≤m}.
```

Lemma 4.20. *Algorithm 4.6 outputs a scaled size-reduced pseudo-basis of the module M generated by the input pseudo-basis and preserves the $\mathcal{N}(r_{ii}I_i)$'s. If $M \subseteq R^s$, then it runs in time polynomial in the input bit-length and in* $\log \Delta_K$.

Proof. As the input basis is scaled, the operations performed on the pseudo-bases preserve the spanned module. Further, note that the update of $\mathbf{b}_j$ at Step 4 has no effect on $r_{i'j'}$ for $j' \neq j$ or $j' = j$ and $i' > i$. It transforms r_{ij} into $r_{ij} - x_i r_{ii}$ and $r_{i'j}$ into $r_{i'j} - x_i r_{i'i}$ for $i' < i$. In particular, the new r_{ij} satisfies $\|r_{ij}/r_{ii}\| \leq n \max_{k\leq n} \|r_k\|$. As the r_k's are a LLL-reduced basis of R, we have that $\max_{k\leq n} \|r_k\| \leq (4n)^{n/2}\Delta_K^{1/2}$. This proves the correctness of Algorithm 4.6. The preservation of the $\mathcal{N}(r_{ii}I_i)$'s is direct, as neither the r_{ii}'s nor the I_i's are modified.

Now, assume that $M \subseteq R^s$. Assume that the outer loop is currently at index j. Consider the inner loop iteration indexed by i. Let m_j^{old} and m_j^{new} respectively denote the value of $\max_{i'<j} \|r_{i'j}/r_{i'i'}\|$ at the start and end of this inner loop iteration. We have:

$$m_j^{new} \leq m_j^{old} + \|x_i\| \cdot \max_{i'<j} \|r_{i'i}/r_{i'i'}\|.$$

We have $\max_{i'<j} \|r_{i'i}/r_{i'i'}\| \leq (4n)^n\Delta_K^{1/2}$, because the first $j-1$ columns are size-reduced. Also, we have $\|x_i\| \leq m_j^{old} + (4n)^n\Delta_K^{1/2}$, as $x_i = \lfloor r_{ij}/r_{ii} \rceil_R$. This gives

$$m_j^{new} \leq (1 + (4n)^n\Delta_K^{1/2})m_j^{old} + (4n)^{2n}\Delta_K.$$

Iterating over the $j - 1 \leq m$ values of i, we obtain that m_j always stays bounded from above by $(1 + (4n)^n\Delta_K^{1/2})^m(m_j^{init} + (4n)^n\Delta_K^{1/2})$, where m_j^{init} is the value of $\max_{i'<j} \|r_{i'j}/r_{i'i'}\|$ at the start of the first inner loop iteration. This implies that $\log \|\mathbf{b}_j\|$ always remains below a polynomial in the input size, m and $\log \Delta_K$. If need be, we can recompute the R-factor at every step of the algorithm (rather than updating it), and, by Lemma 2.25, the cost will still be polynomially bounded in the input bit-length and in $\log \Delta_K$. □

We now consider Algorithm 4.7, which is a variant of Algorithm 4.4 that allows us to prove a bound on the bit cost. The only difference (Step 7) is that we call Algorithms 4.5 and 4.6 at every loop iteration of Algorithm 4.4, so that we are able to master the bit-lengths during the execution. Without

loss of generality, we can assume that the pseudo-basis given as input is scaled and size-reduced: if it is not the case, we can call Algorithms 4.5 and 4.6, which will produce a pseudo-basis of the same module, whose bit-length is polynomial in the input bit-length and in $\log \Delta_K$.

Algorithm 4.7 LLL-reduction over K with controlled bit-lengths

Input: A scaled size-reduced pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq m}$ of a module $M \subseteq R^s$.
Output: An LLL-reduced pseudo-basis of M.

1: **while** there exists $i < m$ such that $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1} I_{i+1}) < \mathcal{N}(r_{i,i} I_i)$ **do**
2: Let M_i be the rank-2 module spanned by the pseudo-basis $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$, with $\mathbf{a}_i = (r_{ii}, 0)^T$ and $\mathbf{a}_{i+1} = (r_{i,i+1}, r_{i+1,i+1})^T$;
3: Find $\mathbf{s}_i \in M_i \setminus \{\mathbf{0}\}$ such that $\mathcal{N}(\mathbf{s}_i) \leq \gamma^n \cdot \lambda_1^{\mathcal{N}}(M_i)$;
4: Set $\mathbf{s}_{i+1} = \mathbf{a}_i$ if it is linearly independent with $\mathbf{s}_i$, and $\mathbf{s}_{i+1} = \mathbf{a}_{i+1}$ otherwise;
5: Call the algorithm of Lemma 2.10 with $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ and $(\mathbf{s}_i, \mathbf{s}_{i+1})$ as inputs, and let $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ denote the output;
6: Update $I_i := I'_i$, $I_{i+1} := I'_{i+1}$ and $[\mathbf{b}_i | \mathbf{b}_{i+1}] := [\mathbf{b}_i | \mathbf{b}_{i+1}] \cdot \mathbf{A}^{-1} \cdot \mathbf{A}'$ (where $\mathbf{A} = [\mathbf{a}_i | \mathbf{a}_{i+1}]$ and $\mathbf{A}' = [\mathbf{a}'_i | \mathbf{a}'_{i+1}]$);
7: Update the current pseudo-basis by applying Algorithm 4.5 and then Algorithm 4.6 to it.
8: **end while**
9: **return** $((I_i, \mathbf{b}_i))_{i \leq m}$.

Theorem 4.21. *Assume that Step 3 of Algorithm 4.7 is implemented with some algorithm $\mathcal{O}$ for some parameter γ. Assume that $\alpha_K > \gamma^{2n} 2^n \Delta_K$. Given as input a scaled and size-reduced pseudo-basis of a module $M \subseteq R^s$, Algorithm 4.4 outputs an* LLL*-reduced pseudo-basis of M in time polynomial in the bit-length of the input pseudo-basis, $\log \Delta_K$ and $1/\log(\alpha_K/(\gamma^{2n} 2^n \Delta_K))$.*

Proof. The correctness proof of Theorem 4.17 still holds. The only adaptation needed is to observe that during the execution of Step 7, none of the $\mathcal{N}(r_{ii} I_i)$'s changes. This is provided by Lemmas 4.19 and 4.20. Further, note that the bound on the number of loop iterations is polynomial in the bit-length of the input pseudo-basis and $1/\log(\alpha_K/(\gamma^{2n} 2^n \Delta_K))$. It remains to prove that the bit-lengths of all the quantities occurring during the execution of the algorithm remain sufficiently small.

For this, it suffices to show that the pseudo-bases keep bounded bit-lengths. As Algorithms 4.5, 4.6 and the algorithm from Lemma 2.10 run in polynomial time, the bit-lengths of all quantities manipulated during a loop iteration are polynomially bounded in terms of the run-time of $\mathcal{O}$, $\log \Delta_K$ and the bit-length of the pseudo-basis at the start of the same loop iteration. It therefore suffices to bound the bit-lengths of the pseudo-bases occurring at the start of each loop iteration. At that moment, the pseudo-bases are scaled, so the bit-lengths of the coefficient ideals are polynomial in $\log \Delta_K$. We now focus on the vectors $\mathbf{b}_j \in R^s$.

Note that $\|\mathbf{b}_j\|^2 = \sum_{i \leq j} \|r_{ij}\|^2$. By size-reducedness, we have that $\|\mathbf{b}_j\| \leq \sqrt{n}(4n)^n \Delta_K^{1/2} \max_i \|r_{ii}\|$. As the pseudo-basis is scaled, we have that $\|\mathbf{b}_j\| \leq \sqrt{n}(8n)^n \Delta_K \max_i \mathcal{N}(r_{ii} I_i)^{1/n}$. Now, note that $\max_i \mathcal{N}(r_{ii} I_i)^{1/n}$ never increases during the execution of the algorithm: this is implied by the part of the proof of Theorem 4.17 involving Π. Initially, it is bounded by a polynomial in the input bit-length. Overall, we obtain that at every start of an iteration of the while loop, the quantity $\log \|\mathbf{b}_j\|$ is bounded by a polynomial in the input bit-length and $\log \Delta_K$. Using Lemma 2.23 allows to complete the proof. □

4.4.3 Finding short vectors for the Euclidean norm

By Lemma 4.15 and Theorem 4.21 with $\alpha_k = (1 + c/m) \cdot \gamma^{2n} 2^n \Delta_K$ for a well-chosen constant c, Algorithm 4.7 may be interpreted as a reduction from finding a $2 \cdot (\gamma^{2n} 2^n \Delta_K)^{m-1}$ approximation to a vector reaching $\lambda_1^{\mathcal{N}}$ in rank-m modules, to finding a γ^n approximation to a vector reaching $\lambda_1^{\mathcal{N}}$ in rank-2 modules.

By using Lemma 2.8, we can extend the above to the Euclidean norm instead of the algebraic norm.

Theorem 4.22. *Let $\gamma \geq 1$, assume that $\log \Delta_K$ is polynomially bounded, and assume that a $\mathbb{Z}$-basis of R is known. Then there exists a polynomial-time reduction from solving $SVP_{\gamma'}$ in rank-m modules (with respect to $\|\cdot\|$) to solving SVP_γ in rank-2 modules, where $\gamma' = (2\gamma\Delta_K^{1/(2n)})^{2m-1}$.*

Proof. The reduction consists in first using Algorithm 4.7 with Step 3 implemented using the oracle solving SVP_γ in rank-2 modules. The parameter α_K is set to $(1 + 1/(100m)) \cdot \gamma^{2n} 2^n \Delta_K$. It does not formally achieve the goal of Step 3, but it suffices for the proof of Lemma 4.16 to go through (in the proof of that lemma, we bound $\lambda_1^{\mathcal{N}}(M_i)$ by using a relationship between $\lambda_1^{\mathcal{N}}(M_i)$ and $\lambda_1(M_i)$).

By Theorem 4.21, the reduction runs in in polynomial time. Further, by Lemma 4.15, the output pseudo-basis satisfies $\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq \alpha_K^{m-1} \cdot \lambda_1^{\mathcal{N}}(M)$. By Lemma 2.8 and by definition of α_K, this gives:

$$\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq 2(\gamma^{2n} 2^n \Delta_K)^{m-1} \cdot 2^{n/2} n^{-n/2} \lambda_1(M)^n.$$

Now, an SVP_γ solver for rank-2 modules directly provides an SVP_γ solver for rank-1 module. We hence use our oracle again, on $I_1\mathbf{b}_1$. This provides a non-zero vector $\mathbf{s} \in I_1\mathbf{b}_1 \subseteq M$ such that $\|\mathbf{s}\| \leq \gamma\sqrt{n}\Delta_K^{1/(2n)} \cdot (\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1))^{1/n}$, by Minkowski's theorem. Combining the latter with the above upper bound on $\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1)$ provides the result. □

4.5 Conclusion

As in the previous chapter, the divide-and-swap algorithm for rank-2 modules presented in this chapter should be seen as a theoretical result, showing that the structure of the module lattices might be used to help finding short vectors. However, because it requires a CVP solver in a lattice of dimension at least n^2, this algorithm cannot be used in practice. The main impact of this algorithm is to show that the LLL algorithm can be extended to any number field R, provided that we change the lattice in which we want to solve CVP. More precisely, the LLL algorithm over $\mathbb{Z}$ requires to be able to solve CVP in the lattice $\mathbb{Z}$. When one wants to generalize LLL to a number field R, it then seems natural to require to be able to solve CVP in the lattice R. As we have seen in the introduction of this chapter, this does not seem to be sufficient, however, we have proposed an alternative lattice which can be used to replace $\mathbb{Z}$ and provide an LLL algorithm over R.

This algorithm raises several open questions. First, the lattice L' we introduce has dimension at least n^2, when the number field R has dimension n. One might hope that a lattice of dimension $O(n)$ (or rather $O(\log|\Delta|)$) would be sufficient. However, in the current version of our algorithm, we do not see how the dimension of the lattice L' could be reduced further. Having a lattice L' of smaller dimension would help us implement our algorithm and check the validity of the heuristics (for the moment, a number field of dimension $n = 10$ already implies a lattice L' of dimension ≥ 100, which is close to the maximum dimension we can currently handle when solving CVP). The same kind of open questions as in the previous chapter also appear in this one. Can we prove our new Heuristic 4.5, or at least part of it? Can we use the structure of the lattice L' to speed up CVP computations in it? It would be interesting to consider these questions even for restricted number fields, such as multiquadratic or cyclotomic number fields.

Chapter 5

Graded Encoding Schemes

Since their introduction in cryptographic constructions by Joux in 2000 [Jou00], cryptographic bilinear maps, as provided by pairings on elliptic curves, have enabled the construction of more and more advanced cryptographic protocols, starting with the Identity-Based Encryption scheme of Boneh and Franklin [BF01]. More abstractly, a group equipped with an efficient bilinear map, and on which some discrete-logarithm-like problems are hard (such as the bilinear Diffie-Hellmann problem), provides foundation for a whole branch of cryptography. A natural open question is whether cryptographic bilinear maps can be generalized to degrees higher than 2 while enjoying hardness of generalizations of the Diffie-Hellmann problem. Such hypothetical objects are referred to as *cryptographic Multilinear Maps* (or, for short, MMaps) [BS03]. We still have no candidate constructions for cryptographic multilinear maps, but a variant of it, called *Graded Encoding Schemes* (or, for short, GES) was defined in [GGH13a], together with a candidate construction. From a functionality perspective, the difference between cryptographic multilinear maps and graded encoding schemes is not a problem for many of the applications.

In this chapter, we are going to focus on graded encoding schemes. In a first section, we will give an overview of the main candidates. We will then focus on the GGH13 candidate. In the last section, we will study the security of the GGH13 map against statistical attacks.

This last section corresponds to a joint work with Léo Ducas, which was published in the proceedings of Asiacrypt 2018 [DP18]. The code which was used to perform the experiments described in this chapter is available at

`http://perso.ens-lyon.fr/alice.pellet___mary/code/statistical_leak.sage`

Contents

5.1 Definition and candidates

In this first section, we first formally define graded encoding schemes. We then describe the main candidate GES, and the known attacks against them.

5.1.1 Definitions

A Graded Encoding Scheme (GES) is an encoding scheme over a ring, where the elements are encoded relatively to 'levels'. It is parametrized by a parameter $\kappa > 0$, called the degree of the GES (this parameter somehow represent the maximum degree of a polynomial that can be applied on the encodings). In a symmetric GES, the levels of the encodings are integers, between 0 and κ. In an asymmetric GES, the levels are vectors in $\mathbb{Z}^\kappa$, with non-negative coefficients. Given encodings of elements, it should be possible to publicly perform some computations on them. It should be possible, given two encodings at the same level, to compute an encoding of the sum, still at the same level. Also, given any two encodings, it should be possible to obtain an encoding of the product, at a level which is the sum of the two levels. Finally, there is some level, called maximum level, such that it is possible to publicly check if an encoding at that level is an encoding of zero of not. We now give a formal definition of graded encoding schemes. This definition differs a little from the original definition of [GGH13a], because the usage of GES has changed since this first work. We define both symmetric and asymmetric GES, but in the following, we will mainly consider the asymmetric ones.

Definition 5.1 (κ-Graded Encoding Scheme)**.** An asymmetric (resp. a symmetric) κ-Graded Encoding Scheme over a ring R consists in the following probabilistic polynomial-time algorithms:

- An algorithm $\texttt{Setup}(1^\lambda, 1^\kappa)$ that takes as input the security parameter and the degree of the graded encoding scheme. It outputs a secret key sk and a public key pk.
- An algorithm $\texttt{Enc}(sk, a, \boldsymbol{v})$ that takes as input the secret key sk, a plaintext element a in the plaintext space (included in the ring R), and a level $\boldsymbol{v}$. The level $\boldsymbol{v}$ is a vector in $\mathbb{Z}^\kappa$ with non-negative coefficients (resp. $\boldsymbol{v}$ is a non-negative integer). It outputs an element u, called encoding of a at level $\boldsymbol{v}$.
- Two algorithm $\texttt{Add}(pk, u_1, u_2)$ and $\texttt{Subtract}(pk, u_1, u_2)$ that take as input the public key pk and two encodings u_1 and u_2 of a_1 and a_2 respectively, at the same level $\boldsymbol{v}$. They output a new encoding at level $\boldsymbol{v}$. For the `Add` algorithm, this encoding is called an encoding of $a_1 + a_2$ at level $\boldsymbol{v}$. For the `Subtract` algorithm, this encoding is called an encoding of $a_1 - a_2$ at level $\boldsymbol{v}$.
- An algorithms $\texttt{Multiply}(pk, u_1, u_2)$ that takes as input the public key pk and two encodings u_1 and u_2 of a_1 and a_2 at levels $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ respectively. It outputs a new encoding, called an encoding of $a_1 \cdot a_2$ at level $\boldsymbol{v}_1 + \boldsymbol{v}_2$, where the addition corresponds to the usual addition of κ-dimensional vectors (resp. addition of integers).
- An algorithm $\texttt{Zero-test}(pk, u)$ that takes as input the public key pk and an encoding u at the maximum level $\boldsymbol{v}^*$. This maximum level is defined by $\boldsymbol{v}^* = (1, 1, \ldots, 1) \in \mathbb{Z}^\kappa$ (resp. $\boldsymbol{v}^* = \kappa$). It outputs a boolean value.
- (Optional) An algorithm $\texttt{Extract}(pk, u)$ that takes as input the public key pk and an encoding u at the maximum level $\boldsymbol{v}^*$. It outputs a bit-string.

The encodings created by the `Enc` procedure are called *fresh* encodings, by opposition to the encodings created by the `Add`, `Subtract` and `Multiply` procedures. We say that a level $\boldsymbol{v}$ is *valid* if all of its coefficients are 0 or 1 (resp. if $\boldsymbol{v} \leq \kappa$). An encoding is said to be *valid* if it is output by the `Enc` procedure, or by the `Add`, `Subtract` or `Multiply` procedures applied to valid encodings, and if its level is valid.

The `Extract` procedure is optional. Some applications like multipartite key exchange need it, whereas others such as obfuscation do not. In this manuscript, we will mainly be interested in obfuscation, hence we will soon forget about the `Extract` procedure.

Functionality. In an ideal world, the algorithms of a κ-graded encoding scheme should satisfy the following functionality conditions.

- Given the public key pk output by the `Setup` algorithm and a valid encoding u of some element a at level $\boldsymbol{v}^*$, algorithm `Zero-test`(pk, u) outputs 1 if $a = 0$ and 0 otherwise.
- Given the public key pk output by the `Setup` algorithm and two valid encodings u_1, u_2 of the same element a at level $\boldsymbol{v}^*$, then `Extract`(pk, u_1) and `Extract`(pk, u_2) produce the same bit-string.

In the real world, we do not have any candidate GES with $\kappa \geq 3$ which satisfy these functionality conditions (and which presumably satisfy some useful security definition). This is because the candidate GES we have use noisy encodings, whose noise increases at each operation `Add`, `Subtract` or `Multiply`. When the noise becomes too large, it becomes impossible to zero-test or extract correctly. To capture the real-world graded encoding schemes in the definition above, we modify the conditions on the `Zero-test` and `Extract` algorithms by requiring that they are correct only with overwhelming probability and if the input encodings are obtained by adding a limited number of fresh encodings. Observe that the number of multiplications is always bounded by the degree of the encodings, hence we just need to bound the number of additions. Also, because of the noise in the encodings, we cannot ask for perfect correctness of the `Zero-test` and `Extract` algorithms, but we still require that the probability of failure should be negligible. We obtain the following conditions, for functionality of a κ-graded encoding scheme in the real world (the parameter m is to be chosen depending on the applications). Let u_1, u_2 be any valid encodings of the same element a at level $\boldsymbol{v}^*$, such that we can develop u_1 and u_2 as a sum of at most m monomials in the fresh encodings. Functionality of the κ-GES holds if

- given the public key pk output by the `Setup` algorithm and u_1 as above, algorithm `Zero-test`(pk, u_1) outputs 1 with overwhelming probability if $a = 0$ and 0 with overwhelming probability otherwise (where the probability is taken over the randomness used in the `Enc` algorithm, when generating the fresh encodings that were used in u_1).
- given the public key pk output by the `Setup` algorithm and u_1, u_2 as above, then `Extract`(pk, u_1) and `Extract`(pk, u_2) output the same bit-string with overwhelming probability over the randomness used in the `Enc` algorithm.

Security. A GES is best viewed as a mathematical object, such that some computational problems related to it are presumably intractable. The [GGH13a] article suggested such a problem called Multilinear Decision Diffie Hellman (MDDH), which generalizes the bilinear decisional Diffie Hellman problem for bilinear maps. In the case of symmetric graded encoding schemes, this problem can be stated as follows.

Definition 5.2 (Multilinear Decision Diffie-Hellman (MDDH) for symmetric κ-GES)**.** Given $\kappa + 1$ encodings $\{u_i\}_{0 \leq i \leq \kappa}$ of randomly chosen plaintext elements $\{a_i\}_{0 \leq i \leq \kappa}$ at level $\boldsymbol{v} = 1$, it should be computationally hard to distinguish between an encoding of $\prod_{i=0}^{\kappa} a_i$ at level κ and an encoding of a uniformly chosen value b at level κ.

Observe that the requirement that there are (at least) $\kappa + 1$ encodings is necessary. Indeed, thanks to the properties of a graded encoding scheme, anyone can publicly multiply up to κ level-one elements, and obtain an encoding at level κ. The conjectured hardness of this problem was used in [GGH13a] to construct one-round multipartite key exchange between $\kappa + 1$ users in the following way. Each user i chooses a random element a_i. It then publishes an encoding u_i of a_i at level 1. It also creates an encoding of a_i at level 0 which it keeps secret. Given the other encodings u_j of a_j at level 1 for $j \neq i$, user i then multiplies them all to create an encoding of $\prod_{j \neq i} a_j$ at level κ. It finally multiplies this encoding by its secret level-0 encoding of a_i to obtain an encoding of $\prod_j a_j$ at level κ, and extract a secret key from it using the `Extract` procedure.

This protocol, however, requires each party i to be able to create an encoding at level 1. This cannot be done by the `Enc` procedure, as it would require all users to know the secret key of the multilinear map. Instead, some authority should publicly make available some encodings at level 1 together with their plaintext value. For re-randomization reasons, the authority also provides multiple encodings of 0 at level 1. Each user i can then create an encoding by secretly adding/subtracting some of these

public encodings and re-randomizing it by adding a random subset of the encodings of zero. However, this means that the MDDH problem should remain hard even in the presence of these public pairs of plaintext/encodings of level 1.

As we shall see below, the three main candidate graded encoding schemes are known to be insecure when encodings of zero at level 1 are known. Hence, the assumption that MDDH is hard is not used a lot currently.

Another hardness assumption related graded encoding schemes could come from the constructions of obfuscators, which are the main applications of GES today. There is no simple hardness assumptions related to graded encoding schemes that would be known to imply obfuscation, but it is known that in some 'ideal multilinear map model'[1] (first defined in [BR13]), obfuscation is possible. The ideal multilinear map model is a model where the adversary only has some black box access to the multilinear map. More precisely, it is given handles to the encodings, and some oracle maintains a table linking these handles to the real encodings. The adversary can ask for addition, subtraction and multiplication of encodings, and obtain new handles corresponding to the resulting encodings. It can also ask to zero-test a handle, and receives the output of the zero-test procedure on the corresponding encoding. It has been proven that obfuscation is possible in this ideal multilinear map model [BR14, BGK+14]. One can then wonder whether a graded encoding scheme can be used to instantiate this model. This is however known to be impossible [BR14, BGK+14], because in the ideal multilinear map model, it is possible to achieve virtual black box obfuscation, a primitive which is known to be impossible to achieve [BGI+01].

The conclusion of this discussion is that there is currently no good one-fits-all hardness assumption for graded encoding schemes. Their security should rather be considered for each construction using them. For example, in Section 5.3.2, we define a specific framework using the GGH13 graded encoding scheme, and we define what we consider is an attack in this framework.

Below, we list some security requirements that arise in most of the applications using graded encoding schemes (they are necessary, but usually not sufficient).

- The encodings at level different from 0 in the symmetric case and different from $(0, 0, \ldots, 0)$ in the asymmetric case should hide the corresponding plaintexts.
- It should not be possible to zero-test encodings at a level which is not the maximum level.
- It should not be possible to meaningfully add/subtract encodings at different levels.

On the other hand, it is possible that the graded encoding scheme has more functionalities than the ones given above. For instance, in the graded encoding schemes we are going to describe in the next section, one can publicly multiply an encoding by a ring element, and obtain an encoding of the product at the same level. This is not known to have any impact on the security of the graded encoding scheme, for their current use in cryptographic constructions.

Multilinear maps. Graded encoding schemes are often called multilinear maps, however, these objects are not exactly the same. We give here the definition of a κ-cryptographic multilinear map (or MMap for short).

Definition 5.3 (κ-Cryptographic Multilinear Map [BS03])**.** For $\kappa + 1$ cyclic groups $G_1, \ldots, G_\kappa, G_T$ (written multiplicatively) of the same order p, a κ-multilinear map $e : G_1 \times \cdots \times G_\kappa \to G_T$ is a map with the following properties:

- It is linear with respect to each variable, that is for any $(g_1, \ldots, g_\kappa) \in G_1 \times \cdots \times G_\kappa$, for any $\alpha \in \mathbb{Z}_p$ and for any $1 \leq j \leq \kappa$, we have
$$e(g_1, \ldots, g_j^\alpha, \ldots, g_\kappa) = e(g_1, \ldots, g_j, \ldots, g_\kappa)^\alpha.$$
- It is non-degenerate, i.e., if for all $1 \leq i \leq \kappa$ we take g_i a generator of G_i, then $e(g_1, g_2, \ldots, g_\kappa)$ is a generator of G_T. This condition is there to avoid the trivial multilinear map that sends everything to 1.

[1] The ideal multilinear map model is typically applied to graded encoding schemes instead of multilinear maps. This is because the term multilinear map is commonly used to refer to graded encoding schemes.

- It is efficiently computable.

If the groups $G_1, \ldots, G_\kappa$ are the same, then the multilinear map is said to be symmetric. If the groups are different, then the multilinear map is said to be asymmetric.

There is some analogy between asymmetric MMap and asymmetric GES. Indeed, if we define $\boldsymbol{v}_i$ to be the vector $(0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{Z}^\kappa$ with a 1 in the i-th position, then we can see the elements g_i^α (of the MMap) as encodings of α at level $\boldsymbol{v}_i$ (of the GES). We can add two encodings at the same level $\boldsymbol{v}_i$, and multiply encodings at levels $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\kappa$. This analogy is not perfect, and there are two main differences between multilinear maps and graded encoding schemes. The first one is that in a multilinear map, we have to apply the pairing operation on all elements at once. On the contrary, a graded encoding scheme allows to multiply two elements, then add this new element with another, and then multiply it again. In a multilinear map, we can perform additions only in the groups G_i and G_T. With the language of graded encoding schemes, it means that we can add encodings only at levels $\boldsymbol{v}_i$'s (containing only 0's except for one 1), or at the maximum level $\boldsymbol{v}^*$. Another difference is that the encodings in a graded encoding scheme are not necessarily unique. Many encodings can correspond to the same plaintext, whereas with a multilinear map the encoding function is bijective. This is why we sometimes require an extraction procedure for graded encoding schemes. The extraction algorithm makes it possible to extract a canonical bit string from different encodings of the same element.

In this thesis, we will consider only graded encoding schemes. As it is commonly done in obfuscation, we will sometimes refer to graded encoding schemes as 'multilinear maps'.

5.1.2 Candidates

There are three main candidate graded encodings schemes. The first one was proposed in 2013 by Garg, Gentry and Halevi [GGH13a]. This graded encoding scheme relies on lattice techniques, and uses elements that are polynomials (for both plaintexts and encodings). After this first candidate was published, Coron, Lepoint and Tibouchi [CLT13] proposed new candidate graded encoding scheme, using integers instead of polynomials, hence supposedly more efficient. The general ideas of the CLT13 scheme are very similar to the ones of [GGH13a]. Two years later, in 2015, Gentry, Gorbunov and Halevi [GGH15] proposed a third candidate. This candidate is not truly a graded encoding scheme, in that it does not allow a user to perform any addition/multiplication it wants on the encodings. Rather, the encodings of the GGH15 scheme come with a graph, and one can only multiply encodings that are adjacent in this graph. However, for most applications, this restricted graded encoding scheme is sufficient. The main interest of this third candidate is that it comes closer to have a security proof based on standard lattice assumptions. It is still only a candidate graded encoding scheme, meaning that it has no security proof based on standard assumptions. However, slight variants of it can be used to obtain cryptographic primitives with security based on the plain LWE problem [WZ17, GKW17].

A fourth graded encoding scheme was recently proposed by Ma and Zhandry [MZ18]. Their construction builds upon the CLT13 graded encoding scheme, but adds more safeguards in order to prevent known attacks against the CLT13 scheme. More precisely, under some assumption called 'vector-input branching program un-annihilatability assumption', they show that their graded encoding scheme resists a large class of attacks, including all known attacks against the CLT13 map.

Attacks. The first three candidates mentioned above are known to be subject to a quite large number of attacks, depending on the context in which they are used. None of them is completely broken for all uses, but many applications using these candidate graded encoding schemes have been broken. First, when encodings of zero at a level strictly below the maximum level (i.e., an encoding that can be multiplied by another encoding to obtain a top-level encoding) are known, then some part of the secret key of the CLT13 and GGH13 schemes can be recovered [CHL$^+$15, HJ16]. In both cases, the part of the secret key recovered by the attacks leads to devastating attacks against constructions using the multilinear map. These attacks impact all constructions using graded encoding schemes with encodings of zero at a small level. One such example was the multipartite key exchange between $\kappa + 1$ users described above, when using the GGH13 or CLT13 map. The GGH15 map was designed so that it was possible to construct a multipartite key exchange from it, without revealing low-level encodings of zero. However, this multipartite key exchange was also broken by Coron, Lee, Lepoint and Tibouchi in 2016 [CLLT16].

When considering all these attacks, the main remaining application of graded encoding schemes is obfuscation. The candidate obfuscators use graded encoding schemes, but they do not need to provide the attacker with encodings of zero, except at the maximum level. Hence, the attacks mentioned above do not apply to these constructions. However, new attacks have been developed, and many of the candidate obfuscators using these graded encoding schemes are now broken (see Chapter 6 for more details). All these attacks rely on some weaknesses of the graded encoding scheme underlying the obfuscator construction. The main weakness that is exploited in attacks is the fact that for all the candidate graded encoding schemes, the zero-testing procedure reveals more than one bit of information when the answer is positive (it usually also outputs a ring element, which leaks some information about some secret parameters of the scheme). By zero-testing many top-level encodings of zero, an attacker may then obtain enough information to break the scheme. These attacks against obfuscators use some specificities of the constructions, and hence do not apply to all obfuscator constructions. In Chapter 6, we discuss in more details the current status of candidate obfuscators.

As mentioned above, the MZ18 construction is designed to resist all known attacks against the CLT13 map. In particular, we do not know of an attack against it even if low-level encodings of zero are provided. This means that this GES can be used for instance to perform multipartite key exchange between $\kappa+1$ users. However, as a compensation for increasing the security, the MZ18 scheme is much less efficient than the CLT13 one. It somehow mimics an obfuscator built upon the CLT13 map, whose functionality would be to compute a graded encoding scheme. The authors can then use techniques developed in obfuscation to protect their GES against known attacks. The consequence of this is that it would be counter-productive to use the MZ18 map for applications such as obfuscation, as the same security guarantee can already be achieved more efficiently from the CLT13 map.

In this thesis, we will focus on the GGH13 construction. In this chapter, we will define it, and describe some statistical leakage that can be obtained in some cases. In the next chapter, we will focus on the main remaining application of the GGH13 multilinear map: obfuscation.

5.2 The GGH13 multilinear map

We describe in this section the asymetric variant of the GGH13 multilinear map construction [GGH13a]. We then explicit the size of the parameters that have to be used for functionality. Finally, we discuss its security.

5.2.1 The GGH13 construction

Let us first describe the GGH13 construction, in its asymmetric setting. As mentioned earlier, the GGH13 multilinear map is in fact a graded encoding scheme. It allows to encode elements of a ring. Anybody can then homomorphically perform additions and multiplications on these elements, under some constraints. It is also possible to publicly test if an encoding at the maximum level encodes zero.

Setup. In all this chapter, and whenever the GGH13 multilinear map is used, the ring R will be instantiated as a power-of-two cyclotomic ring $R = \mathbb{Z}[X]/(X^n+1)$, with n a power of two. The field K is then instantiated as the fraction field of R, i.e., the cyclotomic field $\mathbb{Q}[X]/(X^n+1)$. The GGH13 multilinear map encodes elements of the ring R, modulo a small secret element $g \in R$. The plaintext space will be $R_q := R/(qR)$ for some modulus $q \in \mathbb{Z}$. Recall that for $x \in R$, we write $[x]$ the class of the element x in R_q. This notation will be very useful in this chapter, as we will need to make a clear distinction between elements in R_q and their representatives in R.

On input $(1^\lambda, 1^\kappa)$, where λ is the security parameter and κ is the degree of the multilinear map, the `Setup` algorithm generates the following parameters:

- an integer n which is a power of 2, and hence the ring $R = \mathbb{Z}[X]/(X^n+1)$;
- a (small) element g in R. We let $I = gR$ denote the ideal generated by g in R;
- a (large) positive integer q (which defines $R_q := R/(qR)$ and $[x]$ as the class of x modulo q);
- invertible elements $[z_i] \in R_q^\times$, for $1 \leq i \leq \kappa$, chosen uniformly at random in $R_q^\times$;

- a zero-testing parameter $[p_{zt}] = [hz^*g^{-1}]$ where $[z^*] = [\prod_{1\leq i\leq\kappa} z_i]$ and h is a random element in R, generated according to a Gaussian distribution of standard deviation approximately $\sqrt{q}$.

We detail in Section 5.2.2 the size of the parameters described above (we will choose them to ensure the correctness of the scheme). The public key is composed of (n, q, κ, p_{zt}), while the parameters $(h, g, \{z_i\}_i)$ form the secret key.

Encoding of an element. The GGH13 multilinear map allows to encode cosets of the form $a + I$ for some element a in R. Let $\boldsymbol{v} \in (\mathbb{Z}_{\geq 0})^\kappa$ be a vector of size κ. An encoding of the coset $a + I$ at level $\boldsymbol{v}$ is an element of R_q of the form

$$u = \left[(a + rg) \cdot z_{\boldsymbol{v}}^{-1}\right]$$

where $[z_{\boldsymbol{v}}] = [\prod_{i,\boldsymbol{v}[i]=1} z_i]$ and $a+rg$ is a small element in the coset $a+I$. We recall that $\boldsymbol{v}$ is called the level of the encoding, and that $\boldsymbol{v}$ is said to be valid if and only if its coefficients are in $\{0, 1\}$. Honest use of the GGH13 map should only produce encodings at a valid level. We abuse notation by saying that u is an encoding of a (instead of an encoding of the coset $a + I$).

On input a plaintext element $a + I$, a level $\boldsymbol{v} \in \{0, 1\}^\kappa$ and the secret key $(h, g, \{z_i\}_i)$, the `Enc` algorithm does the following. It first samples an element $a' = a+rg$ according to a Gaussian distribution over the coset $a+I$. Then it outputs $u = \left[a' \cdot z_{\boldsymbol{v}}^{-1}\right]$. The Gaussian distribution used to sample a' should be centered in 0 but is not necessarily spherical. We denote by $\Sigma_{\boldsymbol{v}} \in K_{\mathbb{R}}$ its squared parameter (which will be almost equal to its variance, see Section 2.5.2) and we describe in Section 5.3.3 the different choices of $\Sigma_{\boldsymbol{v}}$ we found in the literature.

Operations on encodings. If u_1 and u_2 are two encodings of elements a_1 and a_2 at the same level $\boldsymbol{v}$ then $u_1 + u_2$ is an encoding of $a_1 + a_2$ at level $\boldsymbol{v}$, and $u_1 + u_2$ is an encoding of $a_1 - a_2$ at level $\boldsymbol{v}$.

If u_1 and u_2 are two encodings of elements a_1 and a_2 at levels $\boldsymbol{v}$ and $\boldsymbol{w}$, then $u_1 \cdot u_2$ is an encoding of $a_1 \cdot a_2$ at level $\boldsymbol{v} + \boldsymbol{w}$. These properties hold for any levels $\boldsymbol{v}, \boldsymbol{w}$, even non valid ones. Observe that during an addition/subtraction operation, the euclidean norm of the numerator of the encodings is multiplied by at most two, while it can be squared during a multiplication.

We also note that the GGH13 map allows to multiply an encoding by a known small plaintext element. Indeed, if u is an encoding of a at level $\boldsymbol{v}$ and b is a (small) ring element, then $b \cdot u$ is an encoding of ab at level $\boldsymbol{v}$. The noise is multiplied by b, hence the requirement for b to be short.

Zero-testing. The zero-testing parameter allows us to test if an encoding u at level $\boldsymbol{v^*}$ is an encoding of zero. On input an encoding u and the zero-testing parameter p_{zt}, the `Zero-test` algorithm computes

$$[w] = [u \cdot p_{zt}].$$

If w is small compared to q (the literature typically requires its coefficients to be less than $q^{3/4}$), the algorithm outputs 1 (meaning that u in an encoding of zero). Otherwise, it outputs 0.

To justify the correctness of this algorithm, let us consider $u = [(a+rg)(z^*)^{-1}]$ a top-level encoding. When we multiply u by the zero-testing parameter, we obtain $[u \cdot p_{zt}] = [(a + rg)hg^{-1}]$. Now, if $a = 0$, then we have $[u \cdot p_{zt}] = [rh]$, with both r and h small compared to q.[2] The parameters of the GGH13 map are chosen to ensure that $\|rh\|_2$ is smaller than $q^{3/4}$. On the other hand, if $a \neq 0$, then we have g^{-1} which appears in the product. It is large, so $[u \cdot p_{zt}]$ is likely to be large too. It can be proven that in this case, the smallest representative of $[up_{zt}]$ in R will never have an euclidean norm smaller than $q^{3/4}$ (see [GGH13a, Section 4.1] for more details).

5.2.2 Size of the parameters and correctness

The size of the parameters of the GGH13 multilinear map depends on the applications we want to use it for. The main difference between the diverse instantiations of the GGH13 map will be the choice of q. Below, we give the size recommended in the original article [GGH13a], and we discuss about the choice of q, depending on the application.

[2]The modulus q will be chosen such that for any allowed top-level encoding of zero, the noise r of this encoding is small compared to q.

- The dimension n of R should be taken such that $n = \Omega(\kappa\lambda^2)$, where λ is the security parameter of the scheme. Taking a lower bound in λ^2 was the original choice of [GGH13a] to avoid some lattice attacks. It was reduced to $n = \Omega(\kappa\lambda\log(\lambda))$ in [LSS14]. However, considering the recent sub-exponential algorithms to solve the principal ideal problem [BF14,BEF+17], it should be increased back to $\Omega(\kappa\lambda^2)$. Also, to prevent recent attacks on the NTRU problem [ABD16, CJL16, KF17], the parameter n should satisfy $n = \Omega(\lambda(\log q)^2)$. When q is set as $\log q = O(\kappa)$ (which is the case in most constructions), the two conditions can be satisfied by taking $n = \Omega(\kappa^2\lambda^2)$.

- The secret element g is sampled using a Gaussian distribution, with rejection, such that $\|g\| = O(n)$ and $\|1/g\| = O(n^2)$.

- The secret element h is sampled using a centered Gaussian distribution of parameter $\sqrt{q}$, so that $\|h\| = \Theta(\sqrt{n}\cdot\sqrt{q})$. In [GGH13a, Section 6.4], the authors suggest to sample h according to a non spherical Gaussian distribution instead of a spherical one.

- In the original GGH13 scheme, the modulus q was chosen to be greater than $2^{8\kappa\lambda}\cdot n^{O(\kappa)}$. This bound came from the re-randomisation procedure used originally to publicly generate level-1 encodings. This setting was shown to be insecure (because it uses encodings of zero), and the current constructions using the GGH13 map do not use this re-randomization technique anymore. So, in the current applications, the modulus q is usually chosen to be as small as possible such that the zero-test is correct. To determine this optimal value, we should compute an upper bound M on the Euclidean norm of the numerator of encodings that are allowed to be zero-tested. Then q can be set as $q = M^4n^{14}$ (or the smallest prime larger than M^4n^{14} if one wants a prime modulus). This way, for any allowed top-level encoding of zero $u = [rg(z^*)^{-1}]$, with $\|rg\| \leq M$, we have $\|rh\| \leq n\cdot\|rg\|\cdot\|h\|\cdot\|g^{-1}\| \leq n^{3.5}\cdot M\cdot q^{1/2} \leq q^{3/4}$ by choice of q. The value of M (and so the choice of q), is highly dependent on the application for which the GGH13 map is used. It depends in particular on the number of additions and multiplications that can be performed honestly on fresh encodings. In Section 5.3.2, we will give explicit values for M and q in the setting we will be considering.

  In most applications of the GGH13 map, the setting requires a user to be able to perform $\Omega(\kappa)$ multiplication of fresh encodings, where κ is usually polynomial in the security parameter λ. This means that $M = 2^{\Omega(\lambda)}$ and so q should be exponential in λ. This will be the case for most of the candidate obfuscators we are going to describe in Chapter 6. In Section 5.3.2 however, we will be interested in cases where M is polynomial in λ, and so q can be set as small as polynomial in λ. Having a small value of q has two main interests. First, it gives more efficient protocols. Second, it may be considered more secure, as some attacks against the GGH13 map [ABD16, CJL16, KF17] require a large modulus. We will however see in Section 5.3 that the GGH13 map with a small modulus seems more vulnerable to statistical attacks.

5.2.3 Security of the GGH13 map

As already discussed in Section 5.1, the GGH13 map suffers from different attacks, depending on the context in which it is used. There are two main categories of attacks against the GGH13 map, the zeroizing attacks, and the NTRU attacks.

NTRU attacks. The NTRU attacks rely on the fact that the quotient of two encodings of the GGH13 map at the same level has an NTRU structure, that is, it is of the form $f/g \bmod q$ where both f and g are small. It has been shown in [ABD16, CJL16] that if the modulus q is large enough compared to the dimension n (roughly $q = 2^{\sqrt{n}}$), then it is possible to recover f/g in K from $f/g \bmod q$ in polynomial time. This means that we can recover multiples of the numerator of encodings of the GGH13 map, which usually breaks any application using the GGH13 map. The recent attack of Kirchner and Fouque [KF17] improves these attacks by exploiting the fact that in the GGH13 map, we are given a lot of NTRU samples $f_i/g \bmod q$ with the same denominator g. These NTRU attacks against the GGH13 map however are not very devastating. The reason for this is that they require the parameter q to be of the order of $2^{\sqrt{n}}$. Hence, by increasing n by a polynomial factor, we can prevent these attacks. There is no circular dependencies here, as q depends on κ and not on n. So by choosing carefully the parameters of the GGH13 map, these attacks can be avoided.

Zeroizing attacks. A more devastating line of attacks against the GGH13 map are called zeroizing attacks. These attacks use the fact that the GGH13 map leaks more information than the one bit 'u is an encoding of zero' when zero-testing a top-level encoding of zero u. Indeed, recall that when we zero-test a top-level encoding of zero $u = [rg/z^*]$, we recover $[rh]$. Now, observe that the product rh is small compared to q, hence, knowing it modulo q means that we know it exactly. So for each successful zero-test, we recover a ring element $rh \in R$, where the noise r depends on the encodings that where used to generate this top-level encoding. This is more than what an ideal graded encoding scheme should reveal. Below, we describe different zeroizing attacks, that use this extra information to try to recover secret informations of the GGH13 map.

A first observation, is that with enough successful zero-tests, we obtain many ring elements $r_i h$, which are all multiples of h. Hence, we heuristically hope that, with a few of them, we can recover the ideal $\langle h \rangle$ (this is true in practice). This can be done in all constructions using the GGH13 multilinear map with top-level encodings of zero. We will use it in Chapter 6 to mount a quantum attack against some obfuscators.

Recall that we mentioned in Section 5.1 that the GGH13 map was not secure if low-level encodings of zero were provided. We can now give the idea of why this is the case. Assume we have two encodings of zero that can be multiplied to obtain a top-level encoding. Because the numerators of the two encodings were multiples of g, then the top-level encoding of zero we obtain is of the form $[rg^2/z^*]$. Hence, if we zero-test it, we recover rgh. If we can create different $r_i gh$, then, as previously, we can heuristically recover the ideal $\langle gh \rangle$. By dividing by the ideal $\langle h \rangle$ we can then recover the ideal $\langle g \rangle$. All this was already known by the authors of the GGH13 map (see [GGH13a, Section 6.3.1]), but they did not know how to use it for an attack. Hu and Jia [HJ16] showed in 2015 that the knowledge of $\langle g \rangle$ can in fact be used to break the security of the multipartite key exchange based on the GGH13 map: an attacker that sees the transcript of the key exchange can reconstruct the secret key.

Observe that the attack above can be extended even if we do not have low-level encodings of zero, as long as we have sufficiently many pairs of plaintext/low-level encodings. Indeed, assume that we have two plaintexts a_1, a_2 and the corresponding encodings u_1, u_2 at the same level $\boldsymbol{v}$. Then, the encoding $a_1 u_2 - a_2 u_1$ is an encoding of zero at level $\boldsymbol{v}$,[3] and the previous attack using encodings of zero can be applied. Since the attack of Hu and Jia, it is considered insecure to use the GGH13 map with low-level encodings of zero, or with known pairs of plaintext/low-level encodings.

Even when the GGH13 map is used without low-level encodings of zero, as it is the case in obfuscation construction, other attacks have been derived. The first one was described by Miles, Sahai and Zhandry in 2016 [MSZ16], and called annihilation attack. They used the fact that the ring elements recovered after zero-testing have some structure that is known by the adversary. Using this structure, they are able to recover the ideal $\langle g \rangle$, and can use it to break the security of the obfuscators. Generalisations of this attack were proposed by [CGH17, ADGM17], also recovering the ideal $\langle g \rangle$ to break the scheme. It is now believed that recovering a non-zero multiple of g can be considered a break of the GGH13 map, as recovering the ideal $\langle g \rangle$ seems to be quite devastating for all the constructions using the GGH13 map.

All the zeroizing attacks described above are algebraic attacks. They use the algebraic structure of the noise to derive polynomials that are multiples of g and then recover the ideal $\langle g \rangle$. These are the main attacks that have been used against the GGH13 map. In the next section, we will focus on a different kind of attacks, which use statistical properties of the noise to try to recover the ideal $\langle g \rangle$. This kind of attacks has not been used before against the GGH13 map, except in the cryptanalytic survey made by the authors of the GGH13 map (see [GGH13a, Section 6.3.2]). In this survey, the authors conclude that their naive construction could be subject to statistical attacks, and propose a counter-measure to try to prevent them.

These statistical attacks are also zeroizing attacks, in that they exploit the information leaked by zero-testing encodings of zero. In order to distinguish them from the previous algebraic attacks we described, we are going to call them statistical zeroizing attacks. We are also going to refer to the other ones as algebraic zeroizing attacks. This terminology is specific to this thesis. Because statistical zeroizing attacks are not common, the term zeroizing attacks is typically used in the literature to refer to algebraic zeroizing attacks.

[3] The plaintexts a_1 and a_2 we know should be somehow small, otherwise the noise of this encoding will be too large.

5.3 Statistical attack on the GGH13 map

This section presents a study of the impact of statistical zeroizing attacks against the GGH13 scheme and some of its variants.

5.3.1 Contribution

As mentioned earlier, there is no simple hardness assumption related to the GGH13 map to study. Hence, in Section 5.3.2, we first start by defining a simple setting using the GGH13 map. We say that an attacker breaks the multilinear map in this setting if it is able to compute a non-zero multiple of g (remember that this is currently considered as a break of the GGH13 map). Once this model is defined, we can study the statistical leakage of the GGH13 map in this model. This leakage will depend on the sampling procedure used in the `Enc` algorithm (i.e., the choice of the matrix $\Sigma_{\boldsymbol{v}}$). Following the nomenclature of [GGH13a, DGG+18], except for the second one that had no clear name, we consider:

1. The simplistic method: the GGH13 MMap without countermeasure [GGH13a, Sec. 4.1]. This method was only given for simplicity of exposition and was already highly suspected to be insecure;
2. The exponential method:[4] this is the countermeasure proposed in [GGH13a, Sec. 6.4] to try to prevent statistical attacks;
3. The conservative method, proposed in [DGG+18] —which we partly revisit to tackle some of its limitations;
4. The aggressive method, proposed in [DGG+18] —we note that this method is specific to the obfuscator construction of [DGG+18], and is not applicable to all constructions over the GGH13 MMap.

These four variants were the only ones we found in the literature, they are described in Section 5.3.3.

After defining the framework and the sampling methods, we analyse in Section 5.3 the leakage that can be obtained for each method. Our analysis shows that Method 3 leads to the same leakage as Method 1. We also prove that with Method 1, a polynomial-time attack can be mounted using the leakage. However, we did not manage to extend the attack to Method 3: while the same quantity is statistically leaked, the number of samples remains too low for the attack to go through completely (due to conditions on the parameters, Method 3 cannot be used to sample as many fresh encodings as we want). On the other hand, we show that the statistical leakage of Method 4 is similar to the one of Method 2: perhaps surprisingly the aggressive method seems more secure than the conservative one.

Finally, having built a better understanding of which information is leaked, we devise in Section 5.3.5 a countermeasure that we deem more adequate than all the above:

5. The compensation method.

This method is arguably simpler, and provides better parameters. More importantly, applying the same leakage attack as above, one only obtains a distribution whose variance is independent of all secrets. We wish to clarify that this is in no way a formal statement of security. The statistical attacks considered in this work are set up in a minimalistic setting, and extensions could exist beyond this minimalistic setting. For example, one could explore what can be done by varying the zero-tested polynomial, or by keeping certain encodings fixed between several successful zero-tests.

5.3.2 Setting and hardness assumption

Simple setting. To study the leakage of the GGH13 multilinear map, we need to make reasonable assumptions on what is given to the adversary. It has been shown in [HJ16] that knowing low-level encodings of zero for the GGH13 multilinear map leads to zeroizing attacks that completely break the scheme. So our setting should not provide any, yet we will provide enough information for some zero-tests to pass.

[4] The naming reflects the fact that this method typically leads to an exponential modulus q.

This setting is inspired by the use of multilinear maps in current candidate obfuscator constructions, and more precisely by the low noise candidate obfuscator of [DGG+18]. Yet, for easier analysis, we tailored this setting to the bare minimum. We will assume that the attacker is given elements that pass zero-test under a known polynomial of degree $\ell = 2$. The restriction $\ell = 2$ can easily be lifted but it would make the exposition of the model and the analysis of the leakage less readable.

More precisely, we fix a number $m > 1$ of monomials, and consider the homogeneous degree-2 polynomial:

$$H(x_1, y_1, \ldots, x_m, y_m) = \sum x_i y_i.$$

We choose a set of *atoms* $\mathcal{A}$ consisting in all levels $\boldsymbol{v} \in \{0,1\}^\kappa$ that have weight exactly 1 or $\kappa - 1$, where the weight of $\boldsymbol{v}$ is the number of its non-zero coefficients (i.e., its ℓ_1 norm). For all $\boldsymbol{v} \in \mathcal{A}$, we let $\tilde{\boldsymbol{v}} = \boldsymbol{v}^* - \boldsymbol{v}$ (we say that $\tilde{\boldsymbol{v}}$ is the complement of $\boldsymbol{v}$). We note that $\mathcal{A}$ is closed by complement. We assume that for each $\boldsymbol{v} \in \mathcal{A}$ of weight 1, the authority reveals encodings $u_{\boldsymbol{v},1}, \ldots, u_{\boldsymbol{v},m}$ at level $\boldsymbol{v}$ of random values $a_{\boldsymbol{v},1}, \ldots, a_{\boldsymbol{v},m}$ modulo I, and encodings $u_{\tilde{\boldsymbol{v}},1}, \ldots, u_{\tilde{\boldsymbol{v}},m}$ at level $\tilde{\boldsymbol{v}}$ of random values $a_{\tilde{\boldsymbol{v}},1}, \ldots, a_{\tilde{\boldsymbol{v}},m}$ modulo I, under the only constraint that

$$H(a_{\boldsymbol{v},1}, a_{\tilde{\boldsymbol{v}},1}, \ldots, a_{\boldsymbol{v},m}, a_{\tilde{\boldsymbol{v}},m}) = 0 \bmod I.$$

We remark that generating almost uniform values $a_{\cdot,\cdot}$ under the constraint above is easily done, by choosing all but one of them at random, and setting the last one to

$$a_{\tilde{\boldsymbol{v}},m} = -a_{\boldsymbol{v},m}^{-1} \sum_{i=1}^{m-1} a_{\boldsymbol{v},i} a_{\tilde{\boldsymbol{v}},i} \bmod I.$$

Definition 5.4 (Successful attack in the simple setting)**.** We say that an attacker breaks this setting if, given one tuple of encodings $(u_{\boldsymbol{v},1}, \ldots, u_{\boldsymbol{v},m}, u_{\tilde{\boldsymbol{v}},1}, \ldots, u_{\tilde{\boldsymbol{v}},m})$ as above for each $\boldsymbol{v} \in \mathcal{A}$ of weight 1, it can create a non-zero multiple of g with non negligible probability. The plaintext elements $a_{\boldsymbol{v},i}$ are arbitrary, conditioned on the fact that $H(a_{\boldsymbol{v},1}, a_{\tilde{\boldsymbol{v}},1}, \ldots, a_{\boldsymbol{v},m}, a_{\tilde{\boldsymbol{v}},m}) = 0 \bmod I$ for all $\boldsymbol{v} \in \mathcal{A}$.

In the weak multilinear map model (specific to the GGH13 map) [MSZ16, GMM+16, DGG+18], we can prove that an attacker that has access to this simple setting of the GGH13 multilinear map cannot recover a non-zero multiple of the secret element g, except with negligible probability. This proof is rather technical and not very informative, hence, we postpone it to Appendix A. The appendix also contains a formal definition of the weak multilinear map model as well as some mathematical tools used in the proof. The proof in the weak multilinear map model means that our simple setting is immune against certain types of algebraic zeroizing attacks. This weak multilinear map model was used to prove security of candidate obfuscators in [GMM+16, DGG+18], as it captures all known algebraic zeroizing attacks, such as the ones of [MSZ16, CGH17].

Choice of q. Now that we have defined our setting, we can specify the value of the modulus q. Recall that we choose it as small as possible so that the zero-testing procedure is correct. In our simple setting, the top-level encodings of zero we want to be able to zero-test are of the form $u_{\max} = H(u_{\boldsymbol{v},1}, u_{\tilde{\boldsymbol{v}},1}, \ldots, u_{\boldsymbol{v},m}, u_{\tilde{\boldsymbol{v}},m})$, for some level $\boldsymbol{v} \in \mathcal{A}$. Let us write $c_{\boldsymbol{v},i}$ the numerator of the fresh encoding $u_{\boldsymbol{v},i}$ for every $\boldsymbol{v} \in \mathcal{A}$ and $i \in \{1, \ldots, m\}$. Then, the numerator of the top-level encoding of zero $u_{\max}$ is $c_{\max} = H(c_{\boldsymbol{v},1}, c_{\tilde{\boldsymbol{v}},1}, \ldots, c_{\boldsymbol{v},m}, c_{\tilde{\boldsymbol{v}},m})$. In order to choose a modulus q, we need to have an upper bound on the Euclidean norm of this numerator. We will obtain this upper bound from upper bounds on the $\|c_{\boldsymbol{v},i}\|$'s, i.e., the norms of the numerators of the fresh encodings.

Depending on the method we will consider in Section 5.3.3, the norm of the numerator of fresh encodings can depend on the dimension κ of the vectors, as well as on the ℓ_1 norm $\|\boldsymbol{v}\|_1$ of the level $\boldsymbol{v}$ of the encoding. In order to unify notations, for a fresh encoding at level $\boldsymbol{v}$ with numerator c, we will write

$$\|c\| = \Theta(n^{\gamma + \eta \cdot \|\boldsymbol{v}\|_1 + \nu K}), \tag{5.1}$$

where γ, η and ν are non-negative real numbers that will depend on the sampling method, and K is a real number such that $\kappa = n^K$. We give in the next section the values of γ, η and ν corresponding to the different sampling methods we consider. When we do not need to focus on the dependence on $\|\boldsymbol{v}\|_1$ and K, we just call $E := \Theta(n^{\gamma + \eta \cdot \|\boldsymbol{v}\|_1 + \nu K})$ the bound above. For each sampling method

described in Section 5.3.3, we choose this bound to be as small as possible under the specific constraints that will arise with the sampling method. The original proposal was setting E and therefore q to be super-polynomial even for bounded degree κ because of the re-randomization technique used for publicly sampling encodings. Since then, attacks using encodings of zero [CGH+15,HJ16] have restricted encodings to be private, allowing polynomially large E. In the following, we will then try to achieve polynomial E when it is possible.

Given this upper bound on the numerator of fresh encodings, we can now compute an upper bound on the numerator of a top-level encoding of zero. First, let us observe that, for any $\boldsymbol{v} \in \mathcal{A}$ and $i \in \{1, \ldots, m\}$, we have $\|c_{\boldsymbol{v},i} \cdot c_{\tilde{\boldsymbol{v}},i}\| \leq O(\sqrt{n} \cdot n^{2\gamma+\eta\cdot\kappa+2\nu K})$, because the weight of the two levels $\boldsymbol{v}$ and $\tilde{\boldsymbol{v}}$ sum to κ. Hence, we obtain the following upper bound on the norm of the numerator of a top-level encoding of zero

$$\|c_{max}\| = \|\sum_{i=1}^{m} c_{\boldsymbol{v},m} \cdot c_{\tilde{\boldsymbol{v}},m}\| \leq O(m \cdot n^{1/2+2\gamma+\eta\cdot\kappa+2\nu K}).$$

Finally, recall from Section 5.2.2 that a sufficient value for the modulus q is $q = M^4 n^{14}+1$, where M is an upper bound on the norm of the numerators of the top-level encodings of zero. Hence, we can take

$$q \geq m^4 \cdot n^{16+8\gamma+4\eta\kappa+8\nu K}. \tag{5.2}$$

5.3.3 Sampling methods

We describe in this section different sampling methods that can be used to generate the fresh encodings of the GGH13 multilinear map. These are all the methods we found in the literature. The first two methods (simplistic and exponential) were described in [GGH13a]. The conservative and aggressive methods were described in [DGG+18], in order to achieve a polynomial modulus q.

In all the following sampling methods except the first one, one chooses a representative $z_{\boldsymbol{v}} \in R$ of $[z_{\boldsymbol{v}}] \in R_q$ for all $\boldsymbol{v} \in \mathcal{A}$. This representative will not necessarily be the canonical one, with coefficients in $[-q/2, q/2]$. Then, we will take $\Sigma_{\boldsymbol{v}} = \sigma_{\boldsymbol{v}}^2 z_{\boldsymbol{v}} \overline{z_{\boldsymbol{v}}}$, with $\sigma_{\boldsymbol{v}} = \Theta(n^2 \|1/z_{\boldsymbol{v}}\|)$. Using Inequalities (2.9) and (2.10), we can see that $\|1/\sqrt{\Sigma_{\boldsymbol{v}}}\| \leq (1/\sigma_{\boldsymbol{v}}) \cdot n^{1/4} \cdot \|1/z_{\boldsymbol{v}}\|$. Hence, with our choice of $\sigma_{\boldsymbol{v}}$ and the fact that $\|g\| = O(n)$, we obtain

$$\left\| \frac{g}{\sqrt{\Sigma_{\boldsymbol{v}}}} \right\| \leq \sqrt{n} \cdot \|g\| \cdot \left\| \frac{1}{\sqrt{\Sigma_{\boldsymbol{v}}}} \right\| = O\left(\frac{1}{n^{1/4}}\right) = o\left(\frac{1}{\sqrt{\log n}}\right).$$

We can therefore apply Theorem 2.28 to sample the numerators of fresh encodings at level $\boldsymbol{v}$, according to a distribution negligibly close to the Gaussian distribution of parameter $\Sigma_{\boldsymbol{v}}$ and support $a + gR$ (for any $a \in R$). Using tail-cut Gaussian distributions (see Section 2.5.2), we have that if c is the numerator of a fresh encoding, then $\|c\| \leq n\|\sqrt{\Sigma_{\boldsymbol{v}}}\| \leq n^{1.5}\sigma_{\boldsymbol{v}}\|z_{\boldsymbol{v}}\|$ with overwhelming probability. Replacing $\sigma_{\boldsymbol{v}}$ by its definition, this means that we can take

$$E = \Theta(n^{3.5} \cdot \|1/z_{\boldsymbol{v}}\| \cdot \|z_{\boldsymbol{v}}\|). \tag{5.3}$$

We recall that E is an upper bound on the norm of the numerator of any allowed top-level encoding of zero. Hence, in the following methods (except the simplistic one), we will focus on the size of $\|1/z_{\boldsymbol{v}}\| \cdot \|z_{\boldsymbol{v}}\|$ to get a bound on the value of E.

5.3.3.1 The simplistic method

The simplistic method consists in always choosing $\Sigma_{\boldsymbol{v}} \sim 1$, independently of $\boldsymbol{v}$ and $z_{\boldsymbol{v}}$. Using Theorem 2.28, sampling an element according to a distribution negligibly close to $D_{a+gR,\sqrt{\Sigma_{\boldsymbol{v}}}}$ is possible if $\Sigma_{\boldsymbol{v}} = \sigma^2$ for a scalar $\sigma \geq \|g\| \cdot \omega(\sqrt{\log n})$. So by taking $\sigma = \Theta(n^{1+\varepsilon})$ with $\varepsilon > 0$, we have $E = \Theta(\sqrt{n}\sigma) = \Theta(n^{1.5+\varepsilon})$, i.e., $\gamma = 1.5 + \varepsilon$ and $\eta = \nu = 0$.

This method was deemed subject to averaging attacks and hence less secure than the following one in [GGH13a], but the authors claim that their attack attempts failed because all recovered elements were larger that $\sqrt{q}$, and that averaging attacks would need super-polynomially many elements. We explicit an attack, and show below that this attack is possible even for exponential q, as long as E^{ℓ}

remains polynomial: in other words, the presence of the mildly large factor h (of size $\sqrt{q}$) can be circumvented.

5.3.3.2 The exponential method

We present here the countermeasure of [GGH13a, Section 6.4], generalized to the asymetric setting of the GGH13 map, as done in [DGG+18, Section 2.1]. For $1 \leq i \leq \kappa$, set z_i to be the canonical representative of $[z_i]$ in R (with coefficients in the range $[-q/2, q/2]$). Using rejection sampling when choosing z_i, assume that $\|z_i\| \cdot \|1/z_i\| \leq Z$; this is efficient for Z as small as $n^{5/2}$ using [DGG+18], and can even be improved to $Z = n^{3/2}$ using Lemma 5.6 below and its corollary.

For $\boldsymbol{v}$ in $\mathcal{A}$, set $z_{\boldsymbol{v}} = \prod z_i^{v_i}$ over R. Recall that (5.3) gives us: $E = \Theta(n^{3.5}\|1/z_{\boldsymbol{v}}\| \cdot \|z_{\boldsymbol{v}}\|)$. We have $\|z_{\boldsymbol{v}}\| \leq n^{(\|v\|_1-1)/2} \prod_{i\in\boldsymbol{v}} \|z_i\|$ and $\|1/z_{\boldsymbol{v}}\| \leq n^{(\|v\|_1-1)/2} \prod_{i\in\boldsymbol{v}} \|1/z_i\|$. Hence we can take

$$E = \Theta(n^{2.5+\|v\|_1} \cdot Z^{\|v\|_1}) = \Theta(n^{2.5+2.5\|v\|_1}).$$

This means that we have $\gamma = 2.5, \eta = 2.5$ and $\nu = 0$.

Correctness is guaranteed for $q \geq n^{\Omega(\kappa)}$ (because $\eta \neq 0$), and because κ is much larger than the constant degree ℓ in [DGG+18], this is not a satisfying solution, as we aim at decreasing q to polynomial. Two alternatives (conservative and aggressive) are therefore developed in [DGG+18].

5.3.3.3 The conservative method [DGG+18]

The first alternative suggested is to proceed as above, but reducing the $z_{\boldsymbol{v}}$ modulo q, i.e., set $z_{\boldsymbol{v}}$ as the representative of $[\prod z_i^{v_i}]$ with coefficients in $[-q/2, q/2]$. One then ensures, by rejection across all the z_i's together, that $\|z_{\boldsymbol{v}}\| \cdot \|1/z_{\boldsymbol{v}}\| \leq n^{2.5}$ for all $\boldsymbol{v} \in \mathcal{A}$. This leads to $E = \Theta(n^{3.5} \cdot n^{2.5}) = \Theta(n^6)$ (i.e., $\gamma = 6,\ \eta = \nu = 0$) and therefore allows correctness for q as small as $n^{O(\ell)}$, which is polynomial for constant degree ℓ.

Using [DGG+18, Lemma 8] restated as Lemma 5.5 below, the authors conclude that this method is quite inefficient because for the above bound to hold simultaneously for all $\boldsymbol{v} \in \mathcal{A}$ with good probability, n must increase together with κ. Indeed, using Lemma 5.5, we can bound the probability that one of the $z_{\boldsymbol{v}}$ does not satisfy $\|z_{\boldsymbol{v}}\| \cdot \|1/z_{\boldsymbol{v}}\| \leq n^{2.5}$ by $2|\mathcal{A}|/n = 4\kappa/n$. So, if we want this probability to be small (e.g., less than $1/2$) in order for the sampling procedure to be efficient, we should increase n with κ.

Lemma 5.5 (Lemma 8 from [DGG+18])**.** *Let $[z]$ be chosen uniformly at random in R_q and z be its canonical representative in R (i.e., with coefficients in $[-q/2, q/2]$). Then it holds that*

$$Pr\left[\|1/z\| \geq n^2/q\right] \leq 2/n.$$

In the following section, we revisit the conservative method by generalizing this lemma.

5.3.3.4 The conservative method, revisited

In the following lemma, we introduce an extra degree of freedom c compared to the Lemma 5.5, but also improve the upper bound from $O(n^{1-c})$ to $O(n^{1-2c})$.

Lemma 5.6. *Let $[z]$ be chosen uniformly at random in R_q and z be its representative with coefficients between $-q/2$ and $q/2$. Then, for any $c \geq 1$, it holds that*

$$Pr[z = 0 \vee \|1/z\| \geq n^c/q] \leq 8/n^{2c-1}.$$

Corollary 5.7. *Assume that q is prime and $q \geq 2n$. Let $[z]$ be chosen uniformly at random in $R_q^\times$ and z be its representative with coefficients between $-q/2$ and $q/2$. Then, for any $c \geq 1$, it holds that*

$$Pr[\|1/z\| \geq n^c/q] \leq 16/n^{2c-1}.$$

The conditions on q (i.e, q prime and $q \geq 2n$) are used in the proof to ensure that at least half of the elements of the ring R_q are invertible. These conditions could be replaced by the condition "$|R_q^\times| \geq |R_q|/2$".

We can use this corollary to compute the probability that one of the $z_{\boldsymbol{v}}$ does not satisfy $\|1/z_{\boldsymbol{v}}\| \leq n^c/q$ when the $[z_i]$'s are independent and chosen uniformly at random in $R_q^\times$ (assuming that q is prime and $q \geq 2n$). Indeed, the $[z_{\boldsymbol{v}}]$'s are uniform in $R_q^\times$ because they are a product of uniform invertible elements, and, by the union bound, we have

$$\begin{aligned}\Pr\left[\exists \boldsymbol{v} \in \mathcal{A} \text{ s.t. } \|1/z_{\boldsymbol{v}}\| > n^c/q\right] &\leq \sum_{\boldsymbol{v}\in\mathcal{A}} \Pr\left[\|1/z_{\boldsymbol{v}}\| > n^c/q\right] \\ &\leq \frac{16|\mathcal{A}|}{n^{2c-1}}.\end{aligned}$$

If we want this probability to be less than $1/2$, in order to re-sample all the z_i's only twice on average, we should take

$$|\mathcal{A}| \leq \frac{n^{2c-1}}{32}. \tag{5.4}$$

But we also have $\|z_{\boldsymbol{v}}\| \leq \sqrt{n}\|z_{\boldsymbol{v}}\|_\infty \leq \sqrt{n}q$, hence $\|1/z_{\boldsymbol{v}}\| \cdot \|z_{\boldsymbol{v}}\| \leq n^{c+0.5}$. In order to minimize E, we wish to minimize c, under (5.4). By taking the minimal value of c that satisfies this constraint, and recalling that $|\mathcal{A}| = 2\kappa$, we obtain

$$E = \Theta(n^{4.5+K/2}).$$

This means that $\gamma = 4.5,\ \nu = 0.5$ and $\eta = 0$. This revisited conservative method is identical to the original one, except that we improve the encodings size bound E. We also change a little the point of view by fixing n first and then obtaining an upper bound on κ (which will appear because $\nu \neq 0$ in E and we want q to be polynomial in n), while the authors of [DGG+18] first fix κ and then increase n consequently. In the following, we will only focus on the revisited conservative method and not on the original one.

Proof of Lemma 5.6. The proof of this lemma relies on the same ideas as the one of [SS13, Lemma 4.1], but here, the element z is sampled uniformly modulo q instead of according to a Gaussian distribution. Let $[z]$ be chosen uniformly at random in R_q and z be its representative with coefficients between $-q/2$ and $q/2$. Recall that we denote $\sigma_j : K \to \mathbb{C}$ the real/complex embeddings of K in $\mathbb{C}$, with $1 \leq j \leq n$ (we see the real embeddings as complex numbers with the naive inclusion $\mathbb{R} \subset \mathbb{C}$). Because we are working in a power-of-two cyclotomic field, we know that the size of z is related to the size of its embeddings (see Section 2.3.4). Hence, if we have an upper bound on the $|\sigma_j(1/z)|$, we also have an upper bound on $\|1/z\|$. Moreover, the σ_j's are morphisms, so $\sigma_j(1/z) = 1/\sigma_j(z)$, and it suffices to have a lower bound on $|\sigma_j(z)|$.

Let $j \in \{1, \cdots, n\}$, there exists a primitive $2n$-th root of unity ζ such that

$$\sigma_j(z) = \sum_{i=0}^{n-1} a_i \zeta^i,$$

where the a_i's are the coefficients of z, and so are sampled uniformly and independently between $-q/2$ and $q/2$. As ζ is a primitive 2^k-th root of unity for some k, there exists i_0 such that $\zeta^{i_0} = I$, where I is a complex square root of -1. So we can write

$$\sigma_j(z) = a_0 + I a_{i_0} + \tilde{z},$$

for some $\tilde{z} \in \mathbb{C}$ that is independent of a_0 and a_{i_0}. Now, we have that

$$\begin{aligned}\Pr\left[|\sigma_j(z)| < \frac{\sqrt{2}q}{n^c}\right] &= \Pr\left[a_0 + I a_{i_0} \in B(-\tilde{z}, \frac{\sqrt{2}q}{n^c})\right] \\ &\leq \frac{\mathrm{Vol}(B(-\tilde{z}, \frac{\sqrt{2}q}{n^c}))}{q^2} \\ &\leq \frac{8}{n^{2c}},\end{aligned}$$

where $B(-\tilde{z}, \sqrt{2}q/n^c)$ is the ball centered in $-\tilde{z}$ of radius $\sqrt{2}q/n^c$. A union bound yields that

$$\Pr\left[\exists j,\ |\sigma_j(z)| < \frac{\sqrt{2}q}{n^c}\right] \leq n \cdot \frac{8}{n^{2c}} = \frac{8}{n^{2c-1}},$$

which in turns implies

$$\Pr\left[\forall j,\ \left|\sigma_j\left(\frac{1}{z}\right)\right| \leq \frac{n^c}{\sqrt{2}q}\right] \geq 1 - \frac{8}{n^{2c-1}}.$$

To complete the proof, recall that for cyclotomic fields of power-of-two order, Equation 2.6 gives us $\|1/z\| = \sqrt{2/n} \cdot \|\sigma(1/z)\| \leq \sqrt{2} \cdot \max_j(|\sigma_j(1/z)|)$. This gives the desired result.

□

Proof of Corollary 5.7. First, note that sampling $[z]$ uniformly over $R_q^\times$ is the same as sampling $[z]$ uniformly in R_q and re-sampling it until $[z]$ is invertible. We let $U(R_q)$ (resp. $U(R_q^\times)$) denote the uniform distribution in R_q (resp. $R_q^\times$). We then have that

$$\Pr_{[z]\leftarrow U(R_q^\times)}[\|1/z\| \geq n^c/q] = \Pr_{[z]\leftarrow U(R_q)}[\|1/z\| \geq n^c/q \mid [z] \in R_q^\times].$$

Using the definition of conditional probabilities, we can rewrite

$$\Pr_{[z]\leftarrow U(R_q)}[\|1/z\| \geq n^c/q \mid [z] \in R_q^\times] = \frac{\Pr_{[z]\leftarrow U(R_q)}[[z] \in R_q^\times \text{ and } \|1/z\| \geq n^c/q]}{\Pr_{[z]\leftarrow U(R_q)}[[z] \in R_q^\times]}.$$

The numerator of this fraction is less than $\Pr_{[z]\leftarrow U(R_q)}[\|1/z\| \geq n^c/q]$, which is less than $\frac{8}{n^{2c-1}}$ using Lemma 5.6. Further, at least half of the elements of R_q are invertible. Indeed, because q is prime, then we know that R_q is isomorphic to $(\mathbb{F}_{q^{n/k}})^k$, where k is the number of prime factors of $X^n + 1$ modulo q. The number of invertible elements of R_q is then equal to the number of invertible elements of $(\mathbb{F}_{q^{n/k}})^k$, which is $(q^{n/k} - 1)^k$ (an elements is invertible in $(\mathbb{F}_{q^{n/k}})^k$ if and only if all its components are invertible). We conclude that the proportion of invertible elements in R_q is $(1 - \frac{1}{q^{n/k}})^k \geq 1 - \frac{k}{q^{n/k}}$. Because $q \geq 2n \geq 2k$, we finally obtain $1 - \frac{k}{q^{n/k}} \geq 1/2$. Hence $\Pr_{[z]\leftarrow U(R_q)}[[z] \in R_q^\times] \geq 1/2$ and we obtain the desired result

$$\Pr_{[z]\leftarrow U(R_q^\times)}[\|1/z\| \geq n^c/q] \leq \frac{16}{n^{2c-1}}.$$

□

5.3.3.5 The aggressive method

This method was proposed by Döttling, Garg, Gupta, Miao and Mukherjee in [DGG+18] in order to instantiate the GGH13 multilinear map for their obfuscator. This method cannot be used for any set of atoms $\mathcal{A}$, as it relies on the fact that the levels at which we encode fresh encodings have a specific structure. Indeed, for each $\boldsymbol{v} \in \mathcal{A}$, we have either $[z_{\boldsymbol{v}}] = [z_i]$ for some $i \in \{1, \cdots, \kappa\}$ or $[z_{\boldsymbol{v}}] = [z^* \cdot z_i^{-1}]$. Using this remark, the secret elements $[z_i]$ are generated in the following way.

For i from 1 to κ do:

- *sample a uniformly random invertible element $[z_i]$ in R_q. Let z_i be the representative of $[z_i]$ in R with coefficients between $-q/2$ and $q/2$, and $\widetilde{z_i}$ be the representative of $[z_i^{-1}]$ in R with coefficients between $-q/2$ and $q/2$.*
- *until both following conditions are satisfied, re-sample $[z_i]$:*

$$\|1/z_i\| \leq n^3/q \tag{5.5}$$
$$\|1/\widetilde{z_i}\| \leq n/q. \tag{5.6}$$

- *if $i = \kappa$, we also re-sample $[z_i]$ until this third condition is also met*

$$\|1/z^*\| \leq n/q, \tag{5.7}$$

where z^* is the representative of $[\prod_{1\leq i\leq\kappa} z_i]$ with its coefficients between $-q/2$ and $q/2$.

Because we sample the $[z_i]$'s from $i = 1$ to κ, when we generate $[z_\kappa]$ all the other $[z_i]$'s are already fixed, so we can define $[z^*]$.

Note that with this method, we re-sample each z_i an expected constant number of times, independently of κ. Indeed, all $[z_i]$'s for $i \leq \kappa - 1$ are sampled independently. Moreover, the two conditions we want are satisfied except with probability at most $\frac{16}{n}$ for each condition (using Corollary 5.7 with $[z_i]$ and $[z_i^{-1}]$ that are uniform in $R_q^\times$ and with $c = 3$ or $c = 1$). So, applying a union bound, the probability that we have to re-sample $[z_i]$ is at most $\frac{32}{n}$, which is less than $1/2$ if $n \geq 64$. The idea is the same for $[z_\kappa]$ except that we also want $\|1/z^*\|$ to be small. Observe that in this case, all the $[z_i]$'s for $i < \kappa$ are already fixed, so $[z^*]$ only depends on $[z_\kappa]$ and is uniform in $R_q^\times$. Hence this last condition is also satisfied except with probability $\frac{16}{n}$ from Corollary 5.7. Therefore, the probability that the three conditions are met for $[z_\kappa]$ is at least $1/2$ as long as $n \geq 96$.

To conclude, if $n \geq 96$, the procedure described above will sample each $[z_i]$ at most twice on average, independently of the choice of κ. So we can choose κ arbitrarily large and the sampling procedure will take time $O(\kappa) \cdot \text{poly}(n)$.

It remains to choose our representative $z_{\boldsymbol{v}} \in R$ of $[z_{\boldsymbol{v}}] \in R_q$ and to get a bound on $\|1/z_{\boldsymbol{v}}\| \cdot \|z_{\boldsymbol{v}}\|$ for all $\boldsymbol{v} \in \mathcal{A}$, in order to get the value of E. We will show that $\|z_{\boldsymbol{v}}\| \cdot \|1/z_{\boldsymbol{v}}\| \leq n^4$ for some choice of the representative $z_{\boldsymbol{v}}$ we detail below.

First case. If $\boldsymbol{v}$ has weight 1, that is $[z_{\boldsymbol{v}}] = [z_i]$ for some i, then we take $z_{\boldsymbol{v}} = z_i$. With our choice of $[z_i]$, we have that $\|1/z_{\boldsymbol{v}}\| \leq n^3/q$. Also, as $\|z_{\boldsymbol{v}}\|$ has its coefficients between $-q/2$ and $q/2$ we have that $\|z_{\boldsymbol{v}}\| \leq \sqrt{n}q$ and hence $\|z_{\boldsymbol{v}}\| \cdot \|1/z_{\boldsymbol{v}}\| \leq n^{3.5} \leq n^4$.

Second case. If $\boldsymbol{v}$ has weight $\kappa - 1$, then there exists $i \in \{1, \cdots, \kappa\}$ such that $[z_{\boldsymbol{v}}] = [z^* \cdot z_i^{-1}]$. We choose as a representative of $[z_{\boldsymbol{v}}]$ the element $z_{\boldsymbol{v}} = z^* \cdot \widetilde{z_i} \in R$, with z^* and $\widetilde{z_i}$ as above (with coefficients between $-q/2$ and $q/2$). We then have

$$\|1/z_{\boldsymbol{v}}\| = \|1/z^* \cdot 1/\widetilde{z_i}\| \leq \sqrt{n} \cdot \|1/z^*\| \cdot \|1/\widetilde{z_i}\| \leq n^{2.5}/q^2.$$

Further, we have that $\|z_{\boldsymbol{v}}\| = \|z^* \cdot \widetilde{z_i}\| \leq \sqrt{n} \cdot \sqrt{n}q \cdot \sqrt{n}q = n^{1.5}q^2$. This finally gives us

$$\|z_{\boldsymbol{v}}\| \cdot \|1/z_{\boldsymbol{v}}\| \leq n^4.$$

To conclude, this method gives us

$$E = \Theta(n^{7.5}).$$

This means that $\gamma = 7.5$ and both η and ν are zero.

A summary of the different values of γ, η and ν for the different sampling methods can be found in Table 5.1, page 102.

5.3.4 Analysis of the leaked value

We describe in this section the information we can recover using averaging attacks, for each of the sampling methods. We will see that depending on the sampling method, we can recover an approximation of $A(z^*h/g)$, or an approximation of $A(h/g)$ or even the exact value of $A(h/g)$. In order to unify notation, we introduce the leakage $\mathfrak{L}$, which will refer to $A(z^*h/g)$ or $A(h/g)$ depending the method. We explain below what is the value of $\mathfrak{L}$ for the different methods, and how we can recover an approximation of it. In the case of the simplistic method, we also explain how we can recover the exact value of $\mathfrak{L}$ from its approximation and how to use it to mount an attack in our simple setting.

5.3.4.1 Statistical leakage

Let $\boldsymbol{v} \in \mathcal{A}$ be of weight 1. We denote by $[u_{\boldsymbol{v}}]$ the encoding $[H(u_{\boldsymbol{v},1}, u_{\tilde{\boldsymbol{v}},1}, \ldots, u_{\boldsymbol{v},m}, u_{\tilde{\boldsymbol{v}},m})]$. Recall that we have $[u_{i,\boldsymbol{v}}] = [c_{i,\boldsymbol{v}} z_{\boldsymbol{v}}^{-1}]$, where $c_{i,\boldsymbol{v}} = a_{i,\boldsymbol{v}} + r_{i,\boldsymbol{v}} g$ for some $r_{i,\boldsymbol{v}} \in R$. So, using the definition of H and the fact that $[u_{\boldsymbol{v}}]$ passes the zero-test, we can rewrite

$$\begin{aligned}
[u_{\boldsymbol{v}} p_{zt}] &= [H(c_{\boldsymbol{v},1}, c_{\tilde{\boldsymbol{v}},1}, \ldots, c_{\boldsymbol{v},m}, c_{\tilde{\boldsymbol{v}},m})(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})^{-1} \cdot z^* h g^{-1}] \\
&= [H(c_{\boldsymbol{v},1}, c_{\tilde{\boldsymbol{v}},1}, \ldots, c_{\boldsymbol{v},m}, c_{\tilde{\boldsymbol{v}},m}) \cdot h g^{-1}] \\
&= H(c_{\boldsymbol{v},1}, c_{\tilde{\boldsymbol{v}},1}, \ldots, c_{\boldsymbol{v},m}, c_{\tilde{\boldsymbol{v}},m}) \cdot h/g.
\end{aligned}$$

Note that the product of the last line is in R, as it is a product of small elements compared to q. Also, the first term is a small multiple of g so we can indeed divide by g. The division by g here is purely formal, because the first term is equal to g times the noise of the encoding, but it will become important later, when we will average over the zero-tested values. We denote by $w_{\boldsymbol{v}} \in R$ the value above (i.e., the representative of $[u_{\boldsymbol{v}} p_{zt}]$ with coefficients in $[-q/2, q/2]$). The term h/g of the product is fixed, but the first factor $H(c_{\boldsymbol{v},1}, c_{\tilde{\boldsymbol{v}},1}, \ldots, c_{\boldsymbol{v},m}, c_{\tilde{\boldsymbol{v}},m})$ depends on $\boldsymbol{v}$: we can average over it. We now analyze this first factor, depending on the method we choose for generating the fresh encodings of the GGH13 map. We will denote by $Y_{\boldsymbol{v}}$ the random variable $H(c_{\boldsymbol{v},1}, c_{\tilde{\boldsymbol{v}},1}, \ldots, c_{\boldsymbol{v},m}, c_{\tilde{\boldsymbol{v}},m})$.

By definition of the polynomial H, we know that $Y_{\boldsymbol{v}} = \sum c_{i,\boldsymbol{v}} c_{i,\tilde{\boldsymbol{v}}}$. Moreover, all the $c_{i,\boldsymbol{v}}$ are independent when i or $\boldsymbol{v}$ vary, and their variance is negligibly close to $\Sigma_{\boldsymbol{v}}$ (because the conditions of Theorem 2.28 are satisfied). The $c_{i,\boldsymbol{v}} c_{i,\tilde{\boldsymbol{v}}}$ are thus centered random variables of variance $\Sigma_{\boldsymbol{v}} \Sigma_{\tilde{\boldsymbol{v}}}$ (observe that the variance of a product of independent centered variables is the product of their variances) and $Y_{\boldsymbol{v}}$ is a centered random variable of variance $m \Sigma_{\boldsymbol{v}} \Sigma_{\tilde{\boldsymbol{v}}}$ (recall that H is a sum of m monomials). Because $Y_{\boldsymbol{v}}$ is centered, then $\mathbb{E}[w_{\boldsymbol{v}}] = 0$ does not leak any information about secret parameters of the GGH13 map. However, the variance $\mathbb{V}[w_{\boldsymbol{v}}] = \mathbb{E}[A(w_{\boldsymbol{v}})] = m \Sigma_{\boldsymbol{v}} \Sigma_{\tilde{\boldsymbol{v}}} \cdot A(h/g)$ may leak some information about the secret elements h and g of the GGH13 map. One technicality here is that we are given only one variable $w_{\boldsymbol{v}}$ for each choice of $\boldsymbol{v}$, and its variance depends on $\Sigma_{\boldsymbol{v}} \Sigma_{\tilde{\boldsymbol{v}}}$, which might depend on $\boldsymbol{v}$. The main idea to solve this difficulty is to observe that the $\Sigma_{\boldsymbol{v}}$ are also random variables (they depend on the $z_{\boldsymbol{v}}$'s, which are generated according to some probability distribution). Hence, we can compute our empirical variance over the randomness used in the `Enc` algorithm, and on the choice of the denominators $\{z_{\boldsymbol{v}}\}_{\boldsymbol{v} \in \mathcal{A}}$.

The value

$$\mathbb{E}[A(w_{\boldsymbol{v}})] = m \cdot A(h/g) \cdot \mathbb{E}[\Sigma_{\boldsymbol{v}} \Sigma_{\tilde{\boldsymbol{v}}}],$$

where the randomness is taken over the `Enc` algorithm and the choice of the $\{z_{\boldsymbol{v}}\}_{\boldsymbol{v} \in \mathcal{A}}$, is the leak we are going to study below (up to multiplication by a scalar). We will first explicit it, i.e., we will explicit the value of $\mathbb{E}[\Sigma_{\boldsymbol{v}} \Sigma_{\tilde{\boldsymbol{v}}}]$ for all the different sampling methods. Then, we will use Hoeffding's bound to estimate the relative error we can obtain, given the number of samples that are available.

Case 1 (the simplistic method). In this case, we have $\Sigma_{\boldsymbol{v}} = \sigma^2$ for all $\boldsymbol{v} \in \mathcal{A}$, for some $\sigma \in \mathbb{R}$. So $\mathbb{E}[\Sigma_{\boldsymbol{v}} \Sigma_{\tilde{\boldsymbol{v}}}] = \sigma^4$. Let us call $\mu := m\sigma^4 \in \mathbb{R}^+$, then we have $\mathbb{E}[A(w_{\boldsymbol{v}})] = \mu \cdot A(h/g)$.

Before going to the other sampling methods, let us make the following remark. For all the methods except the simplistic one, we have $\Sigma_{\boldsymbol{v}} \Sigma_{\tilde{\boldsymbol{v}}} = \sigma_{\boldsymbol{v}}^2 \sigma_{\tilde{\boldsymbol{v}}}^2 A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})$. Moreover, one can observe that in all these methods, there exists a real σ such that $\sigma_{\boldsymbol{v}} \sigma_{\tilde{\boldsymbol{v}}} = \sigma$ for all $\boldsymbol{v} \in \mathcal{A}$ (in fact, $\sigma_{\boldsymbol{v}}$ only depends on the weight of $\boldsymbol{v}$). Hence, we have that $\mathbb{E}[\Sigma_{\boldsymbol{v}} \Sigma_{\tilde{\boldsymbol{v}}}] = \sigma^2 \mathbb{E}[A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})]$ for some $\sigma \in \mathbb{R}^+$, and so it is sufficient to consider $\mathbb{E}[A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})]$ in the following methods.

Case 2 (the conservative method). For this method, we will see that $\mathbb{E}[A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})]$ is a scalar. Indeed, even if all the $z_{\boldsymbol{v}}$ satisfy $[z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}}] = [z^*]$ in R_q, this is not the case in R. Individually each $z_{\boldsymbol{v}}$ is essentially[5] uniform in the hypercube $[-q/2, q/2]^n$, in particular it is isotropic (see Section 2.5.1 for the definition of isotropic variables). For our analysis, let us treat the $z_{\boldsymbol{v}}$ and $z_{\tilde{\boldsymbol{v}}}$ as random variables in R, that are independent when $\boldsymbol{v}$ varies. The independence assumption is technically incorrect, yet as the only dependences are of arithmetic nature over R_q and that the elements in question are large, one does not expect the correlation to be geometrically visible. This independence assumption is also consistent with numerical experiments (see below). Now, because each of the $z_{\boldsymbol{v}}$ is isotropic (and centered), and by independence of $z_{\boldsymbol{v}}$ and $z_{\tilde{\boldsymbol{v}}}$, we have that $z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}}$ is also isotropic. Let us call $\mu_z := \mathbb{E}\left[A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})\right]$ its variance. Recall that as $z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}}$ is isotropic, we have that μ_z is in $\mathbb{R}^+$. We let $\mu = m\sigma^2 \mu_z \in \mathbb{R}^+$, so that we finally obtain that

$$\mathbb{E}[A(w_{\boldsymbol{v}})] = \mu \cdot A(h/g).$$

Thanks to the isotropy of the $z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}}$, all the variables related to $\Sigma_{\boldsymbol{v}}$ have disappeared, and we are in the same situation as in the simplistic method.

Case 3 (the exponential and aggressive methods). In these methods, unlike the conservative method, the variables $z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}}$ are not isotropic variables anymore and therefore the z's do not "average out".

[5] Up to the invertibility condition in R_q

In the exponential method, the identity $z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}} = z^*$ holds over R (where $z^* = \prod_i z_i \in R$ is a representative of $[z^*]$), hence, $z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}}$ is constant when $\boldsymbol{v}$ varies, and we have

$$\mathbb{E}\left[A(w_{\boldsymbol{v}})\right] = \mu \cdot A(hz^*/g),$$

for some scalar $\mu \in \mathbb{R}^+$.

In the aggressive method, we have $z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}} = z^* \cdot \widetilde{z_i} \cdot z_i$ for some $1 \leq i \leq \kappa$, with z^* the representative of $[z^*]$, z_i the representative of $[z_i]$ and $\widetilde{z_i}$ the representative of $[z_i^{-1}]$ with coefficients in $[-q/2, q/2]$. The element z^* is fixed, but, as in the conservative case, we can view the $\widetilde{z_i} \cdot z_i$ as isotropic variables. Moreover, the z_i's are independent when i varies, and because $\widetilde{z_i}$ only depends on z_i, this implies that the $\widetilde{z_i} \cdot z_i$ are independent. We then have $\mathbb{E}\left[A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})\right] = \mu_z A(z^*)$ for some scalar $\mu_z \in \mathbb{R}^+$. Hence, we have again

$$\mathbb{E}\left[A(w_{\boldsymbol{v}})\right] = \mu \cdot A(hz^*/g),$$

for some scalar $\mu \in \mathbb{R}^+$.

Conclusion on the variance. To conclude, we have argued that in all methods,

$$\mathbb{E}\left[A(w_{\boldsymbol{v}})\right] = \mu \cdot \mathfrak{L}$$

for some scalar $\mu \in \mathbb{R}^+$, where the probabilities are taken over the randomness of the `Enc` algorithm and the random choice of the GGH13 parameters z_i. The leaked variable $\mathfrak{L}$ depends on the sampling method in the following way:

- $\mathfrak{L} = A(h/g)$ for the simplistic and the conservative methods.
- $\mathfrak{L} = A(hz^*/g)$ for the exponential and the aggressive methods.

Observe that in the expectation above, we do not need to use the randomness over the choice of the plaintexts $a_{\boldsymbol{v},i}$. This is because the choice of the plaintext elements has no impact on the expectation and variance of the encodings. It only impacts the support of the distributions. Because here we are not interested in algebraic properties (such as the support of the distribution) but only in statistical properties (such as the expectation and variance of the distribution), the choice of the plaintexts has no impact on our attack.

Now, using the fact that the random variables $A(w_{\boldsymbol{v}})$ are independent for different $\boldsymbol{v} \in \mathcal{A}$ of weight 1, we can compute their empirical mean and Hoeffding's inequality will allow us to bound the distance to the theoretical mean. In the following we assume that we know μ.[6]

Speed of convergence. The attacker computes

$$W = \frac{2}{|\mathcal{A}|} \sum_{\substack{\boldsymbol{v} \in \mathcal{A} \\ \boldsymbol{v} \text{ of weight } 1}} A(w_{\boldsymbol{v}}),$$

the empirical mean of the random variables $A(w_{\boldsymbol{v}})$. This is an approximation of $\mu \cdot \mathfrak{L}$. We know that the coefficients of the random variable $w_{\boldsymbol{v}}$ are less than q, so the coefficients of $A(w_{\boldsymbol{v}})$ are less that nq^2. By applying Hoeffding's inequality in R (Corollary 2.27) with $\varepsilon = 1/n$, $B = nq^2$ and $m = |\mathcal{A}|/2$, we have that $\|W - \mu \cdot \mathfrak{L}\|_\infty < \frac{nq^2\sqrt{8 \ln n}}{\sqrt{|\mathcal{A}|}}$ (except with probability at most $2/n$). As the coefficients of $\mu\mathfrak{L}$ are of the order of nq^2, we have a relative error $\delta < \sqrt{8 \ln n / |\mathcal{A}|}$ for each coefficient of $\mu\mathfrak{L}$. As μ is known, this means that we know $\mathfrak{L}$ with a relative error at most $\sqrt{8 \ln n / |\mathcal{A}|}$. Recall that to apply Hoeffding's inequality, the variables $A(w_{\boldsymbol{v}})$ should be independent. The randomness used by the `Enc` algorithm satisfies the independence condition (each encoding is generated with a fresh randomness independent from the previous ones), but for the conservative and aggressive methods, we also need independence of the $z_{\boldsymbol{v}}$ variables. These variables are not independent, but we report in the next paragraph numerical experiments that show that, for our needs, they behave as independent random variables (with the same speed of convergence as provided by Hoeffding's bound).

[6] The value of the scalar μ can be obtained from the parameters of the multilinear maps. If we do not want to analyze the multilinear map, we can guess an approximation of μ with a sufficiently small relative error by an exhaustive search.

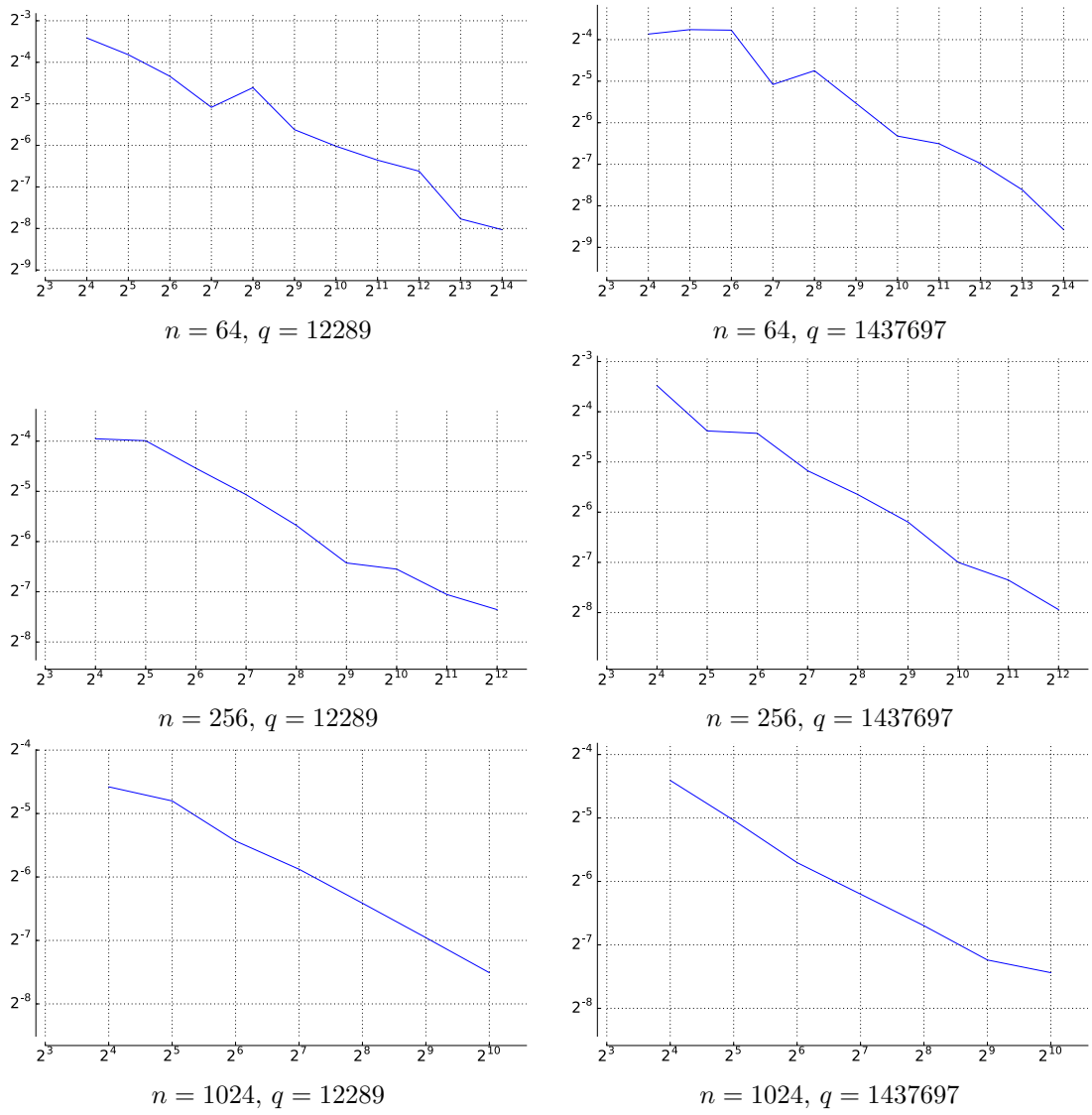

Figure 5.1: Relative precision $\|\varepsilon\|_\infty$ of the empirical mean $\frac{1}{|\mathcal{A}|}\sum_{\boldsymbol{v}\in\mathcal{A}} A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}}) = \mu_z(1+\varepsilon)$ (vertical axis) as a function of $|\mathcal{A}|$ (horizontal axis).

Numerical experiments. We provide here experimental data confirming the heuristic analysis of our attack against the conservative method. More precisely, we study the empirical mean of $A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})$, and the rate of convergence. We computed $\frac{1}{|\mathcal{A}|}\sum_{\boldsymbol{v}\in\mathcal{A}} A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})$ for several values of n, q and $|\mathcal{A}|$ and wrote it as $\mu_z(1+\varepsilon)$, with $\mu_z \in \mathbb{R}^+$ and $\varepsilon \in K_{\mathbb{R}}$. We plotted $\|\varepsilon\|_\infty$ as a function of $|\mathcal{A}|$ in log-log scale, see Figure 5.1.

We observe that $\frac{1}{|\mathcal{A}|}\sum_{\boldsymbol{v}\in\mathcal{A}} A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})$ indeed converges to a constant μ_z in $\mathbb{R}^+$. Furthermore, we observe a slope of about $-1/2$ in log-log scale, confirming that the convergence is as fast as what would be given by the Hoeffding bound if the variables were indeed independent: $\|\varepsilon\|_\infty = f(n)/\sqrt{|\mathcal{A}|}$, for some function f. In fact, it even seems that the function f is decreasing rather than slowly increasing (Hoeffding bound gives $f(n) \leq O(\sqrt{\log n})$). The modulus q seems to have no effect on the relative precision.

5.3.4.2 From the leakage to a complete attack against the GGH13 map

We have seen so far that we can compute an approximation of the leak $\mathfrak{L}$, with relative error $\delta < \sqrt{8 \ln n/|\mathcal{A}|}$. Unfortunately, we cannot directly recover the exact value of $\mathfrak{L}$ from this approximation, because its coefficients are not integers. When $\mathfrak{L} = A(hz^*/g)$, i.e., for the exponential and aggressive methods, we do not know how to use this approximation of $\mathfrak{L}$ to recover the exact value of $\mathfrak{L}$. When $\mathfrak{L} = A(h/g)$, i.e., for the simplistic and conservatives methods, we can circumvent this difficulty. The idea is to transform our approximation of $\mathfrak{L}$ into an approximation of an element $r \in R$, with coefficients that are integers of bit-size logarithmic in the security parameter. Indeed, if we have an approximation of r with error less that $1/2$, we can round its coefficients and recover the exact value of r. Further, we can get such an approximation using a polynomial number of samples because the coefficients we want to recover have logarithmic bit-size. This is what we explain in next subsection. Unfortunately, we will see that for the conservative method, the number of samples we need, in order to be able to round this r to its exact value, is not compatible with the constraint we had on $|\mathcal{A}|$ for being able to generate the $z_{\boldsymbol{v}}$.

We now explain how we can recover the exact value of $A(h/g)$, when $\mathfrak{L} = A(h/g)$ and we have sufficiently many samples. Recovering $A(h/g)$ exactly is a successful attack in our simple setting. Indeed, any denominator of $A(h/g) = (h\overline{h})/(g\overline{g})$ is a non-zero multiple of g.

In the following, we assume that we have an approximation of $A(h/g)$ with relative error $\delta < \sqrt{8 \ln n/|\mathcal{A}|}$ and we want to recover the exact value of $A(h/g)$. Let u be any encoding at level $\boldsymbol{v}^*$ that passes the zero-test (we can take u to be one of the $[u_{\boldsymbol{v}}] = [H(u_{\boldsymbol{v},1}, u_{\tilde{\boldsymbol{v}},1}, \ldots, u_{\boldsymbol{v},m}, u_{\tilde{\boldsymbol{v}},m})]$). We have that $[u \cdot p_{zt}] = c \cdot h/g \in R$ for some small multiple c of g. In particular, the coefficients of c are somehow small and are integers. Using our approximation W of $\mu \cdot A(h/g)$ with relative error δ and the fact that we know μ and $c \cdot h/g$, we can recover an approximation of $A(c)$ with relative error at most $\delta \cdot n^2$ by computing $A(c \cdot h/g) \cdot \mu \cdot W^{-1}$.

The coefficients of $A(c)$ are integers and their magnitudes are less than $m^2 n^2 E^4$. Indeed, we have $c = H(c_{\boldsymbol{v},1}, c_{\tilde{\boldsymbol{v}},1}, \ldots, c_{\boldsymbol{v},m}, c_{\tilde{\boldsymbol{v}},m})$ for some $\boldsymbol{v}$ and we have $\|c_{\boldsymbol{v},i}\| \leq E$ for all $\boldsymbol{v}$'s and i's. So we know that $\|c\| \leq m n^{1/2} E^2$ and we get the desired bound on $\|A(c)\|_\infty$. Hence, if we have an approximation of the coefficients of $A(c)$ with relative error at most $\frac{1}{2m^2n^2E^4}$, the absolute error is less that $1/2$ and we can round the coefficients to recover $A(c)$ exactly. We can then recover $A(h/g)$ exactly by computing $A(c \cdot h/g)/A(c)$.

Putting together the conditions on the parameters, we have $\delta < \sqrt{\frac{8 \ln n}{|\mathcal{A}|}}$ and we want $\delta \cdot n^2 < \frac{1}{2m^2n^2E^4}$ to be able to recover $A(c)$. A sufficient condition for this to be satisfied is $\sqrt{\frac{8 \ln n}{|\mathcal{A}|}} < \frac{1}{2m^2n^4E^4}$, i.e., $|\mathcal{A}| > 32E^8m^4n^8 \ln n$.

To conclude, if $|\mathcal{A}| > 32E^8m^4n^8 \ln n$ and $\mathfrak{L} = A(h/g)$, we obtain an attack against the GGH13 map in our simple setting. In the next sub-section, we compare this constraint to the ones we had for the samplings methods. We will see that for the simplistic method, our constraints are compatible, so we can perform the attack. On the contrary, this will not be the case for the conservative method.

5.3.4.3 Summary of the constraints and conclusion

We summarize in this section the leakage we can obtain and with which precision, depending on the sampling methods presented in Section 5.3.3.

The simplistic method. In this method, we have $\mathfrak{L} = A(h/g)$. Recall that in this case, we can recover the exact value of $\mathfrak{L}$ if $\kappa > 16E^8m^4n^8 \ln n$ (using the fact that $|\mathcal{A}| = 2\kappa$). Also, in this method we had $E = O(n^{1.5+\varepsilon})$, for any $\varepsilon > 0$. Hence, taking $\kappa = \Theta(n^{20+8\varepsilon}m^4 \ln n)$ satisfies the conditions for generating the parameters plus our condition $\kappa > 16E^8m^4n^8 \ln n$. To conclude, when using the simplistic method with some choice of the parameters, we can recover the exact value $A(h/g)$, which means that we have a successful attack against the GGH13 map in our simple setting.

The exponential method. In this method, we have $\mathfrak{L} = A(z^*h/g)$. We can recover an approximation of $\mathfrak{L}$ with relative error at most $\sqrt{\frac{8 \ln n}{|\mathcal{A}|}}$. We do not know if it is possible to recover $\mathfrak{L}$ exactly, or break the GGH13 map in this setting.

The revisited conservative method. In this method, we have $\mathfrak{L} = A(h/g)$. We can recover an approximation of $\mathfrak{L}$ with relative error at most $\sqrt{\frac{8 \ln n}{|\mathcal{A}|}}$ according to our heuristic analysis. While the independence condition between the $A(z_{\boldsymbol{v}} z_{\tilde{\boldsymbol{v}}})$ for applying Hoeffding's bound may not be satisfied, we have seen that this rate of convergence seems correct in practice.

Recall that if $\kappa > 16E^8 m^4 n^8 \ln n$, then we can recover $A(h/g)$ exactly. Also, for the sampling method to work, we need to take $E = \Theta(n^{4.5}\sqrt{\kappa})$. Hence, the condition $\kappa > 16E^8 m^4 n^8 \ln n$ can be rewritten

$$\kappa > \Theta(n^{44} \kappa^4 m^4 \ln n).$$

This condition cannot be satisfied, so we cannot have sufficiently many samples to complete the attack when using this sampling method. All we get is an approximation of $A(h/g)$. Nevertheless, the only thing that prevents the full attack is the size of the parameters we have to choose in order to be able to generate the fresh encodings.

The aggressive method. In this method, we have $\mathfrak{L} = A(z^* h/g)$. We can recover an approximation of $\mathfrak{L}$ with relative error at most $\sqrt{\frac{8 \ln n}{|\mathcal{A}|}}$. We do not know if it is possible to recover $\mathfrak{L}$ exactly.

Conclusion. We give in Table 5.1 a summary of the parameters used for the different sampling methods, and of the resulting leakage. The column 'constraints' specifies possible constraints on the parameters or on the atom set $\mathcal{A}$, that arise when using this sampling method. Recall that due to the correctness bound (5.2), there is always a constraint on the modulus q, so we do not mention it in the column 'constraints'. This constraint on q can be obtained from the columns γ, η and ν, using the formula $\log q \geq 4\log(n)(4 + 2\gamma + 2\nu K + \eta\kappa) + 4\log(m)$.

Sampling method	γ	η	ν	leakage $\mathfrak{L}$	full attack?	constraints
Simplistic [GGH13a]	$1.5 + \varepsilon$	0	0	$A(h/g)$	yes	none
Exponential [GGH13a]	2.5	2.5	0	$A(z^* h/g)$	no	none
Conservative [DGG+18]	6	0	0	$A(h/g)$	no	$n \geq 4\kappa$
Conservative (revisited)	4.5	0	0.5	$A(h/g)$	no	none
Aggressive [DGG+18]	7.5	0	0	$A(z^* h/g)$	no	structure of $\mathcal{A}$
Compensation (Sec. 5.3.5)	$2 + \varepsilon$	0	0	1	no	none

Table 5.1: Summary of the leakage analysis, depending on the sampling method. This includes our new method, sketched in Section 5.3.5. We recall that, according to correctness bound (5.2), the modulus q must satisfy $\log q \geq 4\log(n)(4 + 2\gamma + 2\nu K + \eta\kappa) + 4\log(m)$.

Even if we can only mount a full attack for the simplistic method, we have seen that for every method described above, the leakage is related to some parameters of the GGH13 map that are meant to be kept secret. We would find it more comforting to make the leakage unrelated to secret parameters. In the following section, we propose such a design, which is simple, and leads to smaller parameters.

5.3.5 The compensation method

In this section, we propose a new sampling method which is designed so that the leakage $\mathfrak{L}$ that an attacker can recover, by using the averaging attack described above, reveals no information about secret parameters of the GGH13 map. Nevertheless, we note that even if the attack described above does not apply directly to this method, other averaging attacks may be able to gather information that is supposed to stay secret. An idea could be to fix some encodings and average over the others.

Discussion on design. We have seen that choosing different variance parameters $\Sigma_{\boldsymbol{v}}$ at different levels $\boldsymbol{v}$ can in fact make the leak worse, as the attacker can choose to average them out. We also remark that the parameters $[z_{\boldsymbol{v}}]$ can be publicly re-randomized without affecting anything else, in particular without affecting the variance $\Sigma_{\boldsymbol{v}}$ of the numerator of the encodings. Indeed, we can choose random invertible elements $[\hat{z}_i] \in R_q^\times$, and apply the following transformation to all encodings $e_{\boldsymbol{v}}$ at level $\boldsymbol{v}$, as well as to the zero-testing parameter $[p_{zt}]$:

$$[e_{\boldsymbol{v}}] \mapsto \left[\prod_{i\,:\,\boldsymbol{v}[i]=1} \hat{z}_i^{-1}\right] \cdot [e_{\boldsymbol{v}}], \quad [p_{zt}] \mapsto \left[\prod_{1\le i\le \kappa} \hat{z}_i\right] [p_{zt}].$$

This means that the relation between the variance $\Sigma_{\boldsymbol{v}}$ and the denominators $z_{\boldsymbol{v}}$ can be publicly undone while maintaining functionality.

The compensation method. We therefore proceed to set $\Sigma_{\boldsymbol{v}} = \Sigma$ for all levels $\boldsymbol{v}$, and to choose Σ independently of the $z_{\boldsymbol{v}}$. Recall that the leakage we considered was $\mathbb{V}(w_{\boldsymbol{v}}) = m\Sigma_{\boldsymbol{v}}\Sigma_{\widetilde{\boldsymbol{v}}} \cdot A(h/g)$. Hence, with our choice of $\Sigma_{\boldsymbol{v}}$ this gives us

$$\mathfrak{L} \sim \Sigma^2 \cdot A(h/g). \tag{5.8}$$

We then choose $\Sigma \sim A(g/h)^{1/2}$, ensuring $\mathfrak{L} \sim 1$: the leakage is made constant, unrelated to any secret. We insist nevertheless that, as the previous methods, this method comes with no formal security argument. We also warn that we have not thoroughly explored more general leakage attacks, varying the zero-tested polynomials or keeping some encodings fixed.

It remains to see how one can efficiently sample encodings following this choice. To get tighter bounds, we look at the conditioning number (or distortion) $\delta(\sqrt{\Sigma}) = \frac{\max(\sigma_i(\sqrt{\Sigma}))}{\min(\sigma_i(\sqrt{\Sigma}))}$, where σ_i runs over all embeddings. One easily verifies that, for any $x, y \in K_{\mathbb{R}}$, we have

$$\delta(A(x)) = \delta(x)^2 \tag{5.9}$$

$$\delta(x^k) = \delta(x)^{|k|} \quad \text{for any } k \in \mathbb{R}, \tag{5.10}$$

$$\delta(xy) \le \delta(x)\delta(y). \tag{5.11}$$

If a variable $x \in K_{\mathbb{R}}$ has independent continuous Gaussian coefficients of parameter 1, then its embeddings are (complex) Gaussian variables of parameter $\Theta(\sqrt{n})$, and it holds with constant probability that

$$\forall i, \quad \Omega(1) \le |\sigma_i(x)| \le O(\sqrt{n \log n}). \tag{5.12}$$

Indeed, the right inequality follows from classic tail bounds on Gaussian distribution. For the left inequality, note that $|\sigma_i(x)| \ge \max(|\mathrm{Re}(\sigma_i(x))|, |\mathrm{Im}(\sigma_i(x))|)$, where both the real and imaginary parts are independent Gaussian samples of parameter $\Theta(\sqrt{n})$: either will be smaller than $\Theta(1)$ with probability at most $1/\sqrt{2n}$. By independence, the inequality $|\sigma_i(x)| \le \Theta(1)$ holds with probability at most $1/(2n)$ for each i, and one may conclude by the union bound.

This implies that if x is sampled according to a Gaussian distribution of parameter 1, then $\delta(x) = O(\sqrt{n \log n})$ with constant probability. Moreover, scaling x by a real factor $\sigma > 0$ does not change its conditioning number. Hence, by rejection sampling over h and g, we can ensure that $\delta(g), \delta(h) \le O(\sqrt{n \log n})$ (the rejection probability is constant). We will also scale g to ensure that $\max_i(\sigma_i(g)) = \Theta(n)$. This will imply that

$$\|g\| = O(n) \text{ (see Section 2.3.4)},$$
$$\text{and } \|1/g\| = O(\max_i(\sigma_i(1/g)) = O(1/\min_i(\sigma_i(g))) = O(\sqrt{\log n/n}).$$

Hence, it satisfies the desired conditions for g (recall from Section 5.2.2 that we want $\|g\| = O(n)$ and $\|1/g\| = O(n^2)$). Thanks to the conditions on $\delta(h)$ and $\delta(g)$, we have

$$\delta(\sqrt{\Sigma}) = \delta(A(g/h))^{1/4} \le (\delta(g)\delta(h))^{1/2} \le O(n \log n)^{1/2}.$$

We now define Σ by $\sqrt{\Sigma} = \sigma \cdot A(h/g)^{1/2}$, where the scaling factor $\sigma \in \mathbb{R}_{>0}$ is chosen such that $\min(\sigma_i(\sqrt{\Sigma})) = n \cdot (\log n)^2$. This condition ensures that

$$\begin{aligned}
\|g/\sqrt{\Sigma}\| &\le \sqrt{2} \cdot \max_i(\sigma_i(g/\sqrt{\Sigma})) && \text{(see Section 2.3.4)} \\
&\le \sqrt{2} \cdot \max_i(\sigma_i(g))/\min(\sigma_i(\sqrt{\Sigma})) \\
&= \Theta(n)/\min(\sigma_i(\sqrt{\Sigma})) \\
&= 1/\omega(\log n) && \text{by choice of } \Sigma.
\end{aligned}$$

Hence, we can efficiently sample fresh encodings thanks to Theorem 2.28. Moreover, because $\delta(\sqrt{\Sigma}) = O(\sqrt{n \log n})$, we know that $\|\sqrt{\Sigma}\| = O(\max(\sigma_i(\sqrt{\Sigma}))) = O(n^{1.5+\varepsilon})$ for any $\varepsilon > 0$. We conclude that the numerator of a fresh encoding can be sampled as short as $E = \sqrt{n} \cdot \|\sqrt{\Sigma}\| = O(n^{2+\varepsilon})$, for any $\varepsilon > 0$: the sizes of the numerators of the encodings are barely worse than in the simplistic method, and significantly better than in all other methods.

5.4 Conclusion

In this chapter, we have seen the definition of graded encoding schemes and the example of the GGH13 candidate. We have also presented the first extensive analysis of the statistical leakage that can be obtained from different variants of the GGH13 map, and proposed a variant which seems to leak less information than the ones studied. In the next chapter, we will focus on the security of the GGH13 map when used to construct candidate obfuscators.

Two main open problems are raised by this chapter. The first one is about the leakage obtained for the exponential and aggressive methods. We did not manage to use it to mount any attack against the GGH13 map, but this might still be possible. One way to use the leakage for an attack would be to recover it exactly, as in the simplistic case. However, recovering $A(z^*h/g)$ exactly from an approximation does not seem easy to do, because of the division by g. If the elements were integers, we could recover the fraction from its approximation using continued fractions, but we are working with elements in a number field, and the continued fractions techniques do not seem to translate to this setting. Another direction for attacks could be to try to use the approximation of the leakage without recovering it exactly, but we did not manage to do it either.

Finally, the study of the statistical attack described in this chapter has been done in a specific model, where the attacker can create top-level encodings of zero in a very restricted way. One may wonder whether more complex models can be developed, and with them new statistical attacks. In particular, it would be interesting to know if our compensation method resists more sophisticated statistical attacks.

Chapter 6

Obfuscators

Intuitively, an obfuscator is an algorithm which should render the code of a program unintelligible, while preserving the functionality of the program. On a practical side, companies selling programs are already developing some obfuscation, in order to prevent a user to understand the program and develop a concurrent version of it [XZKL17]. However, practical obfuscation techniques do not come with any formal guarantee that the program is indeed hidden. In this chapter, we will be focusing on the theoretical side of obfuscation. One can define a mathematical object, called an obfuscator, whose specifications should hopefully match the intuition. This object is very desired in theoretical cryptography, as it can then be used to create a lot of cryptographic primitives, with some of them that are not known to be possible otherwise.

In this chapter, we will first define what is an obfuscator, and then present some candidates for general purpose obfuscation. None of the candidate obfuscators that have been proposed so far rely on standard cryptographic hardness assumptions, and we will see that many of these candidate obfuscators suffer from attacks. In particular, we will present a quantum attack which applies to many obfuscators based on the GGH13 multilinear map, some of which were not broken yet. In order to describe this quantum attack, we first describe an abstract obfuscator, capturing many candidates obfuscators. We then present the attack on this abstract obfuscator.

The description of the abstract obfuscator and the quantum attack presented in the last two sections of this chapter correspond to a work which was published in the proceedings of Crypto 2018 [Pel18]. The code used to perform the experiments described in this chapter is available at

`http://perso.ens-lyon.fr/alice.pellet___mary/code/quantum_attack.sage`

Contents

6.1 Introduction

We first start by formally defining obfuscators. We then give a brief overview of the different techniques that have been used to construct candidate obfuscators, and of the attacks that are known against the candidates.

6.1.1 Definition

From the intuitive notion of obfuscation, we would like an obfuscated program to act as a black-box: it should only reveal the input/output behaviour of the program. In other words, we would like that anything that can be learned from the obfuscated program can also be learned by having only a black-box access to the program. This can be formalized by the notion of virtual black-box obfuscation (or VBB obfuscation). In order to formally define the obfuscator, we also need to say how we represent the program to be obfuscated. This is typically done by using circuits. In this thesis, we will consider obfuscators which are defined over a class $\mathcal{C}$ of boolean circuits. The main goal of program obfuscation is to achieve obfuscation for the class $\mathcal{C}$ of all polynomial size circuits (i.e., circuits containing a polynomial number of gates). In this thesis, we will mainly focus of such obfuscators. However, one can also be interested in obfuscators for restricted classes of circuits, for example the class of logarithmic depth circuits, or the class of conjunctions. In the following, we will say that two circuits C_1 and C_2 are *equivalent* (and we will write $C_1 \equiv C_2$), if the sets of inputs of the two circuits are the same, and for any such input x we have $C_1(x) = C_2(x)$ (i.e., C_1 and C_2 compute the same function).

Definition 6.1 (Virtual Black-Box (VBB) obfuscation [BGI+01])**.** A virtual black-box obfuscator $\mathcal{O}$ for the class of circuits $\mathcal{C}$ is a function which takes as input a circuit $C \in \mathcal{C}$ and outputs a circuit $\mathcal{O}(C)$ (not necessarily in $\mathcal{C}$). It should satisfy the following requirements.

- (functionality) For every circuit $C \in \mathcal{C}$, the circuits C and $\mathcal{O}(C)$ are equivalent.
- (efficiency) The obfuscator $\mathcal{O}$ runs in polynomial time in $|C|$, the number of gates of its input circuit C. In particular, this implies that $|\mathcal{O}(C)|$ (the number of gates of $\mathcal{O}(C)$) is polynomially bounded by $|C|$.
- (virtual black-box security) For any probabilistic polynomial time algorithm $\mathcal{A}$, there exists a probabilistic polynomial time algorithm Sim and a negligible function α such that for all circuits $C \in \mathcal{C}$
$$\left|\mathbb{P}\left[\mathcal{A}(\mathcal{O}(C)) = 1\right] - \mathbb{P}\left[\mathrm{Sim}^C(1^{|C|}) = 1\right]\right| \leq \alpha(|C|),$$
where Sim^C means that the algorithm Sim can make black-box queries to the circuit C.

The virtual black-box security requirement means that anything that an adversary can compute from the obfuscated program can also be computed given only a black-box access to the program and the size of the program. This is what we wanted intuitively. However, it has been shown in 2001 by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan and Yang that such a security requirement was not possible to achieve for the class $\mathcal{C}$ of all polynomial size circuits. This does not rule out VBB obfuscation for all classes of circuits, but it still impacts a large number of classes (intuitively, any class containing circuits that can be evaluated on their own descriptions). The main idea of this impossibility result is that any obfuscated circuit $\mathcal{O}(C)$ reveals the code of a circuit computing the same function as C, whereas a black-box access does not. So if for instance the definition of virtual black-box security allowed the attacker to produce a string of bits instead of a single bit, it could output the description of a circuit computing C (by outputting $\mathcal{O}(C)$). On the contrary, the simulator cannot output the description of a circuit computing C, except by making an exponential number of black-box queries to C. This idea can be generalized to an attacker outputting a single bit (as in our definition) by considering circuits that take as input a description of themselves. More formally, the authors of [BGI+01] consider pairs of circuits (C_1, C_2), where $C_1(x)$ is constant (either 0 or 1) for all inputs x describing circuits that compute the same function (i.e., if x and y describe circuits $C_x \equiv C_y$, then $C_1(x) = C_1(y)$). A non-trivial such circuit C_1 can be constructed for instance by choosing two parameters α, β and defining $C_1(x) = 1$ if and only if $C_x(\alpha) = \beta$ (where C_x is the circuit described by x). One can then wonder whether $C_1(C_2) = 1$. An adversary which is given access to

$(\mathcal{O}(C_1), \mathcal{O}(C_2))$ can determine whether $C_1(C_2) = 1$ by computing $\mathcal{O}(C_1)(\mathcal{O}(C_2))$. But a simulator that can only access C_1 and C_2 in a black-box way cannot make such a request, because it does not know any circuit representation of C_2. Finally, to embed the two circuits C_1 and C_2 into a unique circuit C (to match the definition of VBB security), the authors simply consider the circuit C which takes as input bit-strings of size $m+1$ (where C_1 and C_2 take as input bit-string of size m) defined by $C(0\|x) = C_1(x)$ and $C(1\|x) = C_2(x)$.

Virtual black-box security being impossible to achieve for all polynomial size circuit, the authors of [BGI+01] proposed a new security definition for obfuscation, called indistinguishability Obfuscation (or iO for short). An indistinguishability obfuscator satisfies the same functionality and efficiency requirements as before, but its security definition is modified as follows.

Definition 6.2 (Indistinguishability Obfuscation (iO) [BGI+01])**.** An indistinguishability obfuscator $\mathcal{O}$ for the class of circuits $\mathcal{C}$ is a function which takes as input a circuit $C \in \mathcal{C}$ and outputs a circuit $\mathcal{O}(C)$ (not necessarily in $\mathcal{C}$). It should satisfy the following requirements.

- (functionality) For every circuit $C \in \mathcal{C}$, the circuits C and $\mathcal{O}(C)$ are equivalent.
- (efficiency) The obfuscator $\mathcal{O}$ runs in polynomial time in $|C|$, the number of gates of its input circuit C. In particular, this implies that $|\mathcal{O}(C)|$ (the number of gates of $\mathcal{O}(C)$) is polynomially bounded by $|C|$.
- (indistinguishability) For any probabilistic polynomial time algorithm $\mathcal{A}$, there is a negligible function α such that for all *equivalent* circuits $C_1, C_2 \in \mathcal{C}$ of the same size k we have

$$|\mathbb{P}[\mathcal{A}(\mathcal{O}(C_1)) = 1] - \mathbb{P}[\mathcal{A}(\mathcal{O}(C_2)) = 1]| \leq \alpha(k).$$

The definition of indistinguishability obfuscation calls for a few comments. First, as observed by the authors of [BGI+01], if we forget about the efficiency requirement, then iO exists. Indeed, given a circuit C, an obfuscator can output the truth table of the circuit, or even the smallest circuit computing this function (choosing the first one in lexicographic order if there are multiple circuits of the same size). Then, the output of the obfuscator on two equivalent circuits C_1 and C_2 will be the same, and so an attacker cannot distinguish between the two obfuscated circuits. This observation makes the authors of [BGI+01] suggest that there is "some (very slight) hope that this definition is a achievable".

On the other side, one may wonder at first whether, even if they exist, indistinguishability obfuscators are useful. Indeed, it does not correspond anymore to the intuitive notion of obfuscation, where the obfuscated circuit should hide the input circuit. An indistinguishability obfuscator could reveal parts of the circuit we would like to keep secret (for instance a secret key), as long as the obfuscations of equivalent circuits are computationally indistinguishable. It turns out that indistinguishability obfuscation is indeed useful, as was showed by many following works, in constructing many cryptographic primitives from iO. For example, it has been shown that iO could be used to build witness encryption [GGSW13], functional encryption [GGH+13b], deniable encryption [SW14], oblivious transfer [SW14], traitor tracing [BZ17], graded encoding schemes [FHHL18], and so on.

Another way to argue that iO might be useful is to observe that, if we consider the class $\mathcal{C}$ of all polynomial size circuits, then an indistinguishability obfuscator $\mathcal{O}$ achieves *best possible obfuscation* [GR07]. Assume that we have another obfuscator $\mathcal{O}'$ satisfying the functionality and efficiency conditions of iO, but another security definition that we would prefer. We can show that $\mathcal{O}$ reveals less information about the obfuscated circuits than $\mathcal{O}'$. Let $C \in \mathcal{C}$ be a polynomial size circuit. We observe than revealing $\mathcal{O}(\mathcal{O}'(C))$ reveals less information that revealing $\mathcal{O}'(C)$. Indeed, anything an attacker can do with $\mathcal{O}(\mathcal{O}'(C))$, can also be done with $\mathcal{O}'(C)$, by first applying $\mathcal{O}$ to $\mathcal{O}'(C)$. Here, we use the fact that $\mathcal{O}$ is an obfuscator for the class of all polynomial size circuits, and so we can apply it to $\mathcal{O}'(C)$ (because of the efficiency requirement, it is of polynomial size). Now, by the indistinguishability property of $\mathcal{O}$, we know that $\mathcal{O}(\mathcal{O}'(C))$ is indistinguishable from $\mathcal{O}(C)$ (by functionality of $\mathcal{O}'$). Hence, we conclude that anything an adversary can do by knowing $\mathcal{O}(C)$ can also be done given as input $\mathcal{O}'(C)$. This means that $\mathcal{O}$ achieves what we call best possible obfuscation. Notice here that this proof only works because we considered an iO for all polynomial size circuits, so that we can apply $\mathcal{O}$ to the circuit $\mathcal{O}'(C)$. There are examples of classes of circuits for which iO does not achieve best possible obfuscation. One such example is the class of point functions. A point function is described by some bit-string

$y \in \{0,1\}^m$. On input $x \in \{0,1\}^m$, the function outputs 1 if and only if $x = y$ and 0 otherwise. An indistinguishability obfuscator, given as input a bit-string y describing a point function could simply return y. This satisfies the indistinguishability property because each function is uniquely represented, and hence an attacker cannot distinguish between the obfuscation of two circuits computing the same function. On the other hand, we will see in Section 6.1.3 that there exists a VBB obfuscator for point functions under standard assumptions. This obfuscator provably hides the point y, which is not the case of the simple iO described above.

The definition of iO might also be annoying to deal with, because the attacker should choose two circuits C_1 and C_2 which are equivalent. This requirement cannot be checked by the challenger, because being able to determine whether two circuits compute the same function is an NP-hard problem. One can however give an equivalent definition of iO which circumvents this difficulty (the functionality and efficiency requirement of the definition are the same as in Definition 6.2, only the security requirement changes).

Definition 6.3 (Indistinguishability Obfuscation, alternative definition (iO2) [BR14])**.** An iO2 obfuscator $\mathcal{O}$ for the class of circuits $\mathcal{C}$ is a function which takes as input a circuit $C \in \mathcal{C}$ and outputs a circuit $\mathcal{O}(C)$ (not necessarily in $\mathcal{C}$). It should satisfy the following requirements.

- (functionality) For every circuit $C \in \mathcal{C}$, the circuits C and $\mathcal{O}(C)$ are equivalent.
- (efficiency) The obfuscator $\mathcal{O}$ runs in polynomial time in $|C|$, the number of gates of its input circuit C. In particular, this implies that $|\mathcal{O}(C)|$ (the number of gates of $\mathcal{O}(C)$) is polynomially bounded by $|C|$.
- (unbounded simulation) For any probabilistic polynomial time algorithm $\mathcal{A}$, there exists a *computationally unbounded* algorithm Sim and a negligible function α such that for all circuits $C \in \mathcal{C}$

$$\left|\mathbb{P}\left[\mathcal{A}(\mathcal{O}(C)) = 1\right] - \mathbb{P}\left[\mathrm{Sim}^C(1^{|C|}) = 1\right]\right| \leq \alpha(|C|),$$

  where Sim^C means that the algorithm Sim can make black-box queries to the circuit C.

Lemma 6.4 (Lemma 2.9 in [BR14])**.** *Definitions 6.2 and 6.3 are equivalent, i.e., an obfuscator $\mathcal{O}$ is an iO if and only if it is an iO2.*

This lemma in particular implies that a VBB obfuscator (when it exists) is also an indistinguishability obfuscator.

6.1.2 Candidate obfuscators

In this section, we will be interested in candidate indistinguishability obfuscators for all polynomial size circuits. The first such candidate was proposed in 2013 by Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH+13b], and was since followed by many other candidates. These candidates can be sorted into three main categories: branching program obfuscators, circuit obfuscators and obfuscators built from functional encryption. The first two categories are somehow similar. They both construct candidate obfuscators directly, relying on candidate multilinear maps. The third category is somehow different: the guiding principle of these constructions is to try to reduce the construction of iO to the simplest possible primitives, and then propose candidate instantiations of these simple primitives. We give in this section a brief overview of these three categories. In the rest of the chapter we will mainly focus on branching program obfuscators, and, more specifically, on the ones relying on the GGH13 map (hence, their description is a little more detailed). More details can be found in a survey written in January 2018 by Horváth and Buttyán [HB18].

6.1.2.1 Branching program obfuscators

The first candidate iO proposed by Garg et al. in 2013 [GGH+13b] was a branching program obfuscator (we later refer to it as the GGHRSW construction). This means that it takes as input a matrix branching program (see Section 2.6) instead of a circuit. Like circuits or Turing machines, branching programs are a way to represent and compute a function, and this representation of functions was more suited for

the GGHRSW construction than others. Barrington's theorem states that any circuit with logarithmic depth can be efficiently transformed into a branching program with polynomially many matrices that are 5 by 5 permutation matrices. Hence, an iO for the class of all polynomial size branching programs implies an iO for the class NC^1 of logarithmic depth circuits. The authors of [GGH$^+$13b] then proved that any iO for NC^1 can be bootstrapped to an iO for all polynomial size circuits using levelled fully homomorphic encryption, which can be achieved using standard hardness assumptions.

The work of Garg at al. [GGH$^+$13b] hence implies that building iO for branching programs is sufficient to obtain iO for all circuits. The authors also provide the first candidate construction of iO for branching programs. Since then, many other iO candidates for branching programs have been proposed.

Almost all branching program obfuscators rely on graded encoded schemes. Recall that we have three main candidates GES which can be used to build obfuscators: the GGH13 map, the CLT13 map and the GGH15 map. Most of the obfuscators using the GGH13 map can also be instantiated with the CLT13 map and vice versa. However, because of its different structure, the GGH15 map cannot be used in obfuscators built to work with the GGH13 or the CLT13 map. From a security point of view, the known attacks against the candidate branching program obfuscators all rely on weaknesses of the underlying multilinear map, and so differ depending on the chosen multilinear map.

Below, we give a non-exhaustive list of iO candidates for branching programs and attacks. For readability, we draw three separate figures (Figures 6.1, 6.2 and 6.3), one for each underlying multilinear map. As mentioned above, some constructions appear in both Figures 6.1 and 6.2, because they can be instantiated with the GGH13 map and the CLT13 map. In these figures, the references above the axis refer to constructions whereas the ones below the axis refer to attacks.

BP obfuscators using the GGH13 map

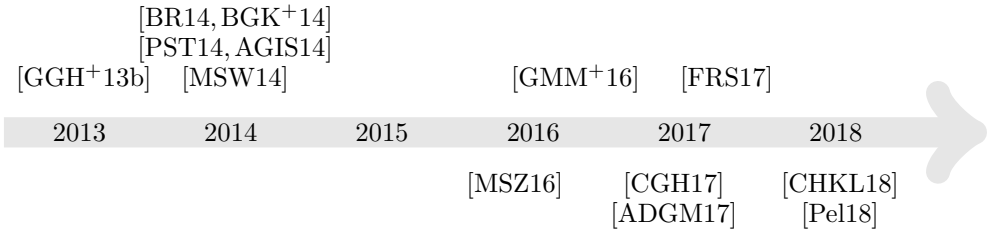

Figure 6.1: History of GGH13-based branching program obfuscators (non-exhaustive)

As mentioned above, the first candidate branching program obfuscator was proposed in 2013 by Garg, Gentry, Halevi, Raykova and Waters [GGH$^+$13b], and was instantiated using the GGH13 map. This construction came with no reduction to standard cryptographic hardness assumptions. This construction was soon followed by many variants, trying to improve both security and efficiency of the GGHRSW construction. First, in 2014, Brakerski and Rothblum [BR14] and Barak, Garg, Kalai, Paneth and Sahai [BGK$^+$14] proposed candidate obbuscators which they proved secure in the *ideal multilinear map model* (the BR14 candidate also requires an extra assumption called *Exponential Time Hypothesis*). Recall from Section 5.1 that the ideal multilinear map model is a model where the adversary does not get the encoded elements, but can only access them via "handles", maintained by the challenger. Pass, Seth and Telang [PST14] then proposed a hardness assumption for graded encoding schemes which they proved implies iO. They also showed that this hardness assumption holds in the ideal multilinear map model. Miles, Sahai and Weiss [MSW14] then proposed a candidate obfuscator proved secure in a model which gives a bit more power to an attacker than the ideal multilinear map model (the main difference being that the attacker can add two encodings at different levels). Finally, Ananth, Gupta, Ishai and Sahai [AGIS14] focused on efficiency and proposed an obfuscator that can obfuscate any matrix branching program (previous candidates took as input branching programs made of permutation matrices). They also showed how this can be used to remove the bootstrapping part of [GGH$^+$13b], and improve efficiency of the obfuscator.

Most of these constructions with proofs in ideal models in fact prove VBB security of the obfuscator (stronger than iO but impossible to achieve), and then use the fact that VBB implies iO to conclude on the iO security of the obfuscators. One of the consequences of these proofs is then that the ideal models considered above cannot be instantiated (as they imply something false). Let us also mention that the ideal multilinear map model (and its variants) used in the above obfuscators is independent of the choice of the multilinear map. The obfuscators can then be instantiated with any candidate multilinear map (i.e., GGH13 of CLT13). This won't be the case anymore for the *weak multilinear map models* discussed below.

After this first line of works, the first attack against candidate iOs was described by Miles, Sahai and Zhandry in 2016 [MSZ16]. This attack relies on some weakness of the underlying GGH13 multilinear map. More specifically, the authors observed that the elements obtained after zero-testing are of the form $\ell_i(x_1, \ldots, x_r) \bmod g$, where the ℓ_i are linear forms that are known by the attacker (and depend on the obfuscated circuit), the x_j are noise terms that are unknown to the attacker (the number r of noise terms is polynomial), and g is the secret parameter of the GGH13 map as defined in Section 5.2. The attack then consists in gathering $r+1$ such zero-tested elements, in order to be able to compute a linear combination of them which annihilates the linear forms ℓ_i (whatever the choices of the x_j's are). Once the ℓ_i are annihilated, we are left with a multiple of g, and we can use it to mount a distinguishing attack. This attack impacts all the previous GGH13-based candidate obfuscators, except the GGHRSW one (i.e., it impacts [BR14, BGK+14, PST14, AGIS14, MSW14]).

A new candidate obfuscator was then proposed by Garg, Miles, Mukherjee, Sahai, Srinivasan, and Zhandry [GMM+16], designed to prevent the MSZ attack. More formally, the authors defined a new idealized model, called the *weak GGH13 multilinear map model*,[1] and proved the security of their candidate iO in this model. This model is designed to capture the MSZ attack, hence the security proof ensures that this attack (or variants of it) cannot be applied to this obfuscator. Again, the authors proved VBB security of their obfuscator, which implies iO security, but also implies that this model cannot be instantiated. Another observation is that this obfuscator can be instantiated with the GGH13 map or the CLT13 map, but its security is only analyzed when instantiated with the GGH13 map. In particular, the weak GGH13 multilinear map model is not a generic model like the ideal multilinear map model: it is defined only when the underlying multilinear map is the GGH13 map.

In 2017, Apon, Döttling, Garg and Mukherjee [ADGM17] extended the MSZ attack. More specifically, the MSZ attack broke iO security of the BP obfuscators by distinguishing between the obfuscation of two equivalent BP. Apon et al. broke the iO security of the full circuit obfuscators by exhibiting two circuits that, once transformed into branching programs and obfuscated, could be distinguished. This attack applies to the same candidate obfuscators as the MSZ one. The same year, Chen, Gentry and Halevi [CGH17] described an attack against the original GGHRSW obfuscator. This attack combines techniques from both the MSZ attack and from attacks that were previously used against the CLT13 map [CLLT17]. Both attacks [CGH17, ADGM17] only work when the corresponding iOs are instantiated with the GGH13 map. A few months later, Fernando, Rasmussen and Sahai [FRS17] proposed a fix for the GGHRSW obfuscator, which prevents the [CGH17] attack (this fix was originally intended for CLT13-based obfuscators).

Finally, in 2018, two "partial" attacks were proposed against GGH13-based BP obfuscation. The first one was proposed by Cheon, Hhan, Kim and Lee [CHKL18]. It impacts the security of the [GGH+13b] and [GMM+16] candidates for specific choices of parameters. This attack can be prevented by increasing the parameters of the iO candidates. The second one is a quantum attack, which I proposed and was published in the proceedings of Crypto 2018. A description of this attack is provided in Section 6.3. It provides an attack against the [GMM+16] obfuscator, but only in the quantum setting.

Table 6.1 summarizes the current state of the art (May 2019) of the candidate branching program obfuscators based on GGH13. In the quantum setting, the only candidate which is still standing is the GGHRSW obfuscator, when fixed by [FRS17]. In the classical setting, the [GMM+16] obfuscator is also standing. The parameters of both of these obfuscators should be chosen large enough to prevent the [CHKL18] attack.

[1] In the literature, this model is typically called "weak multilinear map model" without reference to the GGH13 map. However, it is specific to the GGH13 map, and, as we shall see, similar models have been developed for the CLT13 and GGH15 map. Hence, in this section, to avoid ambiguity, we shall call it "weak GGH13 multilinear map model". A description of this model can be found in Appendix A.

Attacks \ iOs	[GGH$^+$13b]	[BR14,BGK$^+$14,PST14] [AGIS14, MSW14]	[GMM$^+$16]
[MSZ16]		completely broken	
[CGH17]	broken but fixed in [FRS17]		
[CHKL18]	broken for specific choices of parameters	broken for specific choices of parameters	broken for specific choices of parameters
[Pel18]			quantumly broken

Table 6.1: Status of candidate branching program iO based on GGH13 (May 2019)

BP obfuscators using the CLT13 map

Figure 6.2: History of CLT13-based branching program obfuscators (non-exhaustive)

The first candidate obfuscators we have seen in the GGH13 case [GGH$^+$13b,BR14,BGK$^+$14,PST14, AGIS14, MSW14] can also be instantiated by the CLT13 map. Moreover, recall that these obfuscators (expect for the [GGH$^+$13b] one) are proved secure in the ideal multilinear map model (or one of its variants), which is independent from the choice of the underlying multilinear map. On the contrary, the candidate iO from [GMM$^+$16] can be instantiated with the CLT13 map but its security analysis is performed only when instantiated with the GGH13 map, hence we chose not to represent it here.

The first attacks against CLT13-based candidate obfuscators was presented by Coron, Lee, Lepoint and Tibouchi in 2017 [CLLT17]. It impacted all previous candidate obfuscators based on the CLT13 map (i.e., all [GGH$^+$13b, BR14, BGK$^+$14, PST14, AGIS14, MSW14]), when using *single input* matrix branching programs. Recall from Section 2.6 that a branching program is said to be single-input if the choices of the matrices in the execution of the program only depend on one bit of input at each step. The notion of single input branching program is in fact not the best notion to describe the CLLT attack. What the authors really need is *input partitionability* of the branching program. Recall that the function computed by a branching program $\mathbf{A}$ is zero on input x if and only if

$$A_0 \cdot \left(\prod_{i \in [\ell]} A_{i,x[\mathtt{inp}(i)]} \right) \cdot A_{\ell+1} = 0.$$

This branching program is said to be input partitionable if this product of matrices can be split into a product of two terms, which depend on different bits of the input. More formally, we would like that

$$A_0 \cdot \left(\prod_{i \in [\ell]} A_{i,x[\mathtt{inp}(i)]} \right) \cdot A_{\ell+1} = A_{x[\sigma(1),\ldots,\sigma(r)]} \cdot B_{x[\sigma(r+1),\ldots,\sigma(m)]},$$

where σ is a permutation over $\{1,\ldots,m\}$, the integer r is arbitrary between 1 and $m-1$, matrix $A_{x[\sigma(1),\ldots,\sigma(r)]}$ only depends on the bits $\sigma(1),\ldots,\sigma(r)$ of the input x and matrix $B_{x[\sigma(r+1),\ldots,\sigma(m)]}$ only depends on the bits $\sigma(r+1),\ldots,\sigma(m)$ of the input x. For a program to be input partitionable, we will also require that there are two "large" sets[2] $Y \subset \{0,1\}^r$ and $Z \subset \{0,1\}^{m-r}$ such that for all

[2]See the definition of input partitionability in [FRS17] for a definition of "large".

$(y, z) \in Y \times Z$, we have $A_y \cdot B_z = 0$. The definition of input partitionability ensures that we will be able to create many top-level encodings of zero of the form $A_y \cdot B_z$, where we can vary y and z independently. This is the crucial property which is used in the CLLT attack.

Going back to single input/dual input branching programs, one can observe that the dual input BP obfuscators described in [BR14, BGK+14, PST14, AGIS14, MSW14] were not input partitionable, whereas the single-input ones were. Hence, the CLLT attack was applicable only to the single input variants of these iO. One could however imagine dual input BP obfuscators which would be input partitionable, and so vulnerable to the CLLT attack. On the other hand, Fernando, Rasmussen and Sahai developed in [FRS17] a technique to transform any single input branching program into an equivalent single input branching program which is provably not input partitionable. This transformation uses what they call a "tag", which they add to the function to ensure that whenever one tries to modify independently parts of the bits of the input (i.e., varying y and z independently), then the output of the function should not remain equal to 0. This technique hence thwarts the CLLT attack on all the single input obfuscators which were previously broken.[3]

Finally, in 2018, Ma and Zhandry [MZ18] proved that the dual input variant of the [BMSZ16] obfuscator was secure in a weak multilinear map model for the CLT13 map that they defined. This weak CLT13 model is designed to capture the [CLLT17] attack, as well as the other known attacks against the CLT13 map [CHL+15, CGH+15] (not necessarily related to iO). The [BMSZ16] obfuscator was originally designed to obfuscate evasive functions (i.e., functions for which it is hard to find an input which evaluates to zero). However, it can also be used to obfuscate non evasive functions, and the security proof in the weak CLT13 model done in [MZ18] holds for all functions.

To summarize, none the candidate obfuscators above is known to be classically broken when instantiated with the CLT13 map. This holds for both the single input variants of the dual input variants of the obfuscators (using the fix of [FRS17] in the single input case). Additionally, we even have a candidate obfuscator [BMSZ16] which is proven secure in a weak CLT13 multilinear map model [MZ18], a model which captures all known attacks against the CLT13 map. However, the security of these obfuscators only holds in the classical setting. Indeed, the CLT13 map is known to be broken if factorisation can be done efficiently, and so cannot be used in a quantum setting.

BP obfuscators using the GGH15 map

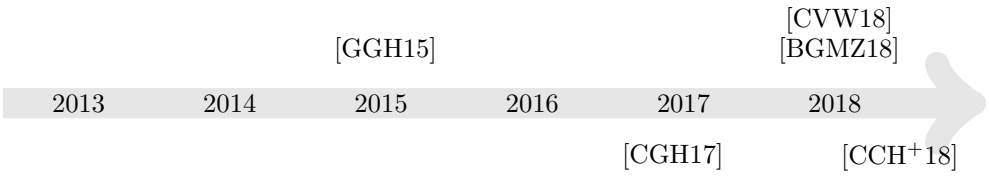

Figure 6.3: History of GGH15-based branching program obfuscators (non-exhaustive)

The GGH15 map was described in 2015 by Gentry, Gorbunov and Halevi [GGH15]. As mentioned earlier, this is a "graph induced" graded encoding scheme, meaning that it comes with a directed acyclic graph, and the levels of the encodings correspond to paths in this graph. One can then add two encodings associated with the same path, or multiply two encodings whose paths come one after the other (i.e. the first one ends where the second one starts). Together with this candidate graded encoding scheme, the authors of [GGH15] also proposed a candidate obfuscator using it. It was quantumly broken in 2017 by Chen, Gentry and Halevi [CGH17]. This quantum attack can also be transformed into a sub-exponential time classical attack, which can be prevented by a polynomial increase of the parameters of the obfuscator. In 2018, two new candidate obfuscators using the GGH15 map were proposed. Chen, Vaikuntanathan and Wee [CVW18] did an extensive study of the attacks against the GGH15 map (by considering previous attacks and developing new ones) and then proposed a candidate iO accordingly. In a similar direction, Bartusek, Guan, Ma and Zhandry [BGMZ18] introduced a weak

[3]Recall from the GGH13 section that the [FRS17] article also prevents the [CGH17] attack. This is because this attack also exploited input partitionability.

multilinear model for the GGH15 map, which captures all known attacks against the GGH15 map. They then proposed a candidate obfuscator which they proved is secure in this weak GGH15 model. A few months later, Cheon, Cho, Hhan, Kim and Lee [CCH$^+$18] presented a statistical zeroizing attack against these candidats obfuscators. This attack was not studied in [CVW18] and is not captured by the weak GGH15 model of [BGMZ18]. However, this attack only applies to specific choice of parameters, and it can be avoided by carefully choosing the parameters of the obfuscators (note that the original choice of parameters of the [BGMZ18] obfuscator already prevented the attack).

To summarize, none of the three candidate obfuscators based in the GGH15 map is known to be broken by a classical computer if the parameters are chosen carefully. In addition, the two recent obfuscators [CVW18, BGMZ18] are also not known to be broken by a quantum computer, and the latter can be proven secure in a weak GGH15 multilinear map model, which captures many but not all known attacks against the GGH15 map.

6.1.2.2 Circuit obfuscators

Circuits obfuscators are similar to branching program obfuscators, except that they obfuscate a circuit directly, without first transforming it into a branching program. As for branching program obfuscators, the circuit obfuscators rely on graded encoding schemes. All the candidates below can be instantiated using either the CLT13 map or the GGH13 map (there is currently no circuit obfuscator using the GGH15 map).

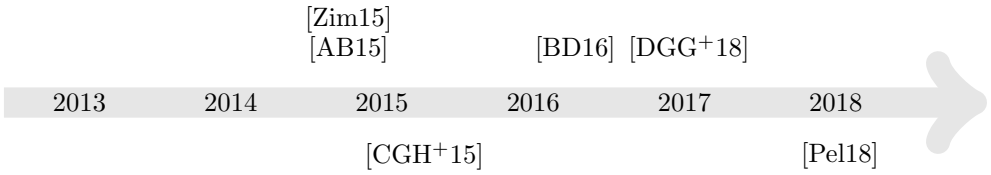

Figure 6.4: History of circuit obfuscators (non-exhaustive)

The first three candidate obfuscators of Figure 6.4 share a very similar structure. Each of these candidate obfuscators is a variant of the same simple obfuscator, with some extra structure designed to achieve different security guarantees. Zimmerman proved virtual black-box security for his obfuscator [Zim15] in the ideal multilinear map model. The obfuscator of Applebaum and Brakerski [AB15] is proved to be iO in a slightly weaker model but at the expense of a loss of efficiency. Finally, the obfuscator of Brakerski and Dagmi [BD16] avoids this loss in efficiency, but assumes sub-exponential hardness of factoring (i.e., it assumes that there exists some $\delta > 0$ such that any algorithm running in time 2^{λ^δ} cannot factor a product of two random primes of λ bits, except with negligible probability). These obfuscators can be instantiated with both the CLT13 map and the GGH13 map. In case of instantiation with the CLT13 map, Coron, Gentry, Halevi, Lepoint, Maji, Miles, Raykova, Sahai and Tibouchi [CGH$^+$15] exhibited an attack against the simple obfuscator common to all the candidates. This attack is not known to extend to the full constructions of these obfuscators.

The last circuit obfuscator candidate, proposed by Döttling, Garg, Gupta, Miao and Mukherjee [DGG$^+$18], also shares the same structure, but is designed to use the GGH13 map with a small modulus (see Section 5.2). This candidate is proved to achieve VBB security in the weak GGH13 multilinear map model. The quantum attack which is described below against GGH13-based branching program obfuscators can be adapted to also impact all the circuits obfuscators above. This provides a quantum attack against all the circuit obfuscator above when instantiated with the GGH13 map (more details on this extension can be found in Appendix B).

To summarize, the four circuit obfuscators above are quantumly broken when instantiated with the GGH13 or the CLT13 map (recall that the CLT13 map is not post quantum). On the contrary, they are not known to be classically broken, neither for the GGH13 map nor for the CLT13 one.

6.1.2.3 Obfuscation from functional encryption

A functional encryption scheme is an encryption scheme where secret keys are associated to functions. A user who is given an encryption of some message m and a secret key sk_f associated to some function f can use its secret key to recover $f(m)$, but should not learn anything more about the message m.

Obfuscation from functional encryption is a different approach to build obfuscation, which was initiated by Lin in 2016 [Lin16]. The objective of this line of works is to reduce the construction of iO to the construction of (hopefully) simpler primitives. The main path followed by all these works is to first reduce iO to functional encryption for circuits of logarithmic depth, and then reduce functional encryption for logarithmic depth circuits to functional encryption for constant degree polynomials, hence the name "obfuscation from functional encryption".

The recent line of works [AJS18, Agr19, LM18] has managed to reduce the construction of iO to the construction of some weak variants of pseudo-random generators (whose name and exact definition differ depending on the articles), computable by a polynomial of degree 2. This research area has been very active recently but is a little outside the subject of this thesis, as it does not involve the GGH13 map or any candidate graded encoding scheme.

6.1.3 Obfuscation for restricted classes of functions

In the previous section, we have seen candidate iOs for the class of all polynomial size circuits. Other approaches have investigated the construction iO for restricted classes of circuits under more standard assumptions. In 2016, Brakerski, Vaikuntanathan, Wee and Wichs [BVWW16] described an obfuscator for conjunction functions, under some variant of RLWE which they called "entropic RLWE". More recently, Wichs and Zirdelis [WZ17] and Goyal, Koppula and Waters [GKW17] concurrently proposed an obfuscator for obfuscating compute-and-compare functions, based on the plain LWE problem. A compute-and-compare function is parametrized by a function f and an element y. On input x, it outputs 1 if and only if $f(x) = y$, and 0 otherwise. This notion includes in particular the classes of conjunctions and of point functions. Let us make a few remarks about these constructions. The first observation is that these constructions use the GGH15 map, but in a way that enables them to obtain a security proof from LWE. One of the key ingredients for this security proof to carry on is that an attacker should not be able to construct top-level encodings of zero. Hence, the obfuscators of [WZ17, GKW17] can only obfuscate compute-and-compare functions with enough min-entropy, so that it is very unlikely that an attacker finds a point where the function outputs 1. The second observation is that while this obfuscator is proven to be VBB secure,[4] this does not contradict the impossibility result of [BGI$^+$01]. This is because we are considering a restricted class of functions, whereas the impossibility result holds for the class of all polynomial size circuits.

6.1.4 Contribution

The objective of the rest of this chapter is to present a quantum attack against candidate obfuscators. This quantum attack impacts the [GMM$^+$16] obfuscator, when instantiated with the GGH13 multilinear map. This candidate obfuscator was not broken yet, and was proven secure in the weak GGH13 multilinear map model (the currently strongest ideal model for GGH13-based obfuscators). As a secondary contribution, our attack also applies to the obfuscators of [BGK$^+$14, PST14, AGIS14, MSW14], which were already broken in classical polynomial time by [MSZ16]. Our attack is still interesting for these obfuscators, as it uses different techniques than those of [MSZ16], and in particular, techniques that are not captured by the weak GGH13 multilinear map model. We note that our attack does not work against the obfuscator of [BR14], while [MSZ16] does.

Overall, we prove the following theorem (informally stated for the moment, see Theorem 6.7 for a formal statement).

Theorem 6.5 (Informal, heuristic)**.** *Let $\mathcal{O}$ be any of the branching program obfuscators in [BGK$^+$14, PST14, AGIS14, MSW14, GMM$^+$16], on single or dual input branching programs, instantiated with the GGH13 multilinear map [GGH13a]. There exist two explicit equivalent branching programs $\boldsymbol{A}$ and $\boldsymbol{A}'$*

[4] It is in fact *distributional* VBB secure, which is weaker than VBB. However, the impossibility result of [BGI$^+$01] also holds for distributional VBB security.

such that $\mathcal{O}(\boldsymbol{A})$ and $\mathcal{O}(\boldsymbol{A}')$ can be distinguished in quantum polynomial time, under some conjecture and heuristic.

We note that the only part of our attack which is quantum is the principal ideal solver of Biasse and Song [BS16]. All the other steps of our attack are classical. Hence, our attack can also be viewed as a (classical) reduction from the iO security of the candidate obfuscators mentioned in Theorem 6.5 to the principal ideal problem. One might then want to use the classical sub-exponential time principal ideal solver of Biasse, Espitau, Fouque, Gélin and Kirchner [BEF+17] to obtain a classical sub-exponential time attack against the above obfuscators. However, the dimension of the cyclotomic ring used in current instantiations on the GGH13 multilinear map is chosen to be at least λ^2 where λ is the security parameter (see Section 5.2.2). This is done to thwart the attacks of [ABD16, CJL16, KF17] over the GGH13 multilinear map, but it also means that the classical variant of the attack described in this chapter is exponential in the security parameter, even when using the sub-exponential time principal ideal solver of [BEF+17]. It is still interesting to note that any future improvement for solving the principal ideal problem will directly imply an improvement for the attack described in this chapter.

Technical overview. The attack described below belongs to the class of mixed-input attacks. A mixed-input attack is an attack in which the attacker does not evaluate honestly the obfuscated circuit, but changes the value of one bit along the computation: for example, if the same bit of the entry is used twice during the computation, the attacker puts it to 1 the first time and to 0 the second time. Recent branching program obfuscators, starting with the one of [BGK+14], prevent mixed-input attacks by relying on the underlying multilinear map and using so-called straddling set systems. The idea of straddling set systems techniques is to choose specific levels for the encodings, to ensure that an attacker who tries to mix the bits of the input will obtain a final encoding which does not have the good level to be zero-tested and provide a useful output. Following this idea, the obfuscators of [PST14, AGIS14, MSW14, GMM+16] also used straddling set systems to prevent mixed-input attacks.

However, straddling set systems only ensure that an attacker cannot mix the inputs of the obfuscated program to obtain a dishonest top-level encoding of zero: it does not prevent an attacker to create a dishonest encoding of zero at a level higher than the top level. In the case where the multilinear map is ideal, this is not a security threat, because the attacker should not be able to test at a level higher than the top level whether it has created an encoding of zero or not. However, this is not the case of the GGH13 multilinear map. Indeed, using recent improvements on the short Principal Ideal Problem [BS16, CGS14a, CDPR16] (abbreviated as sPIP), it has been shown that it is possible to recover in quantum polynomial time the secret element h of the zero-testing parameter of the GGH13 map (see Section 5.2 for more details on the GGH13 map). Recovering this secret element will then allow us to zero-test at a higher level than the one initially authorised.[5] This is the starting point of our mixed-input attack.

As mentioned above, all these candidate obfuscators use straddling set systems, meaning that performing a dishonest evaluation of the branching program outputs an encoding at a forbidden level. However, if we perform two well-chosen dishonest evaluations and take the product of the resulting encodings, we can obtain an encoding whose level is twice the maximal level of the multilinear map. The idea to construct well-chosen dishonest evaluations is to take complementary ones. For instance, assume the first bit of the input is used three times during the evaluation of the branching program. A first illegal computation could be to take this first bit to be equal to 0 the first time it is used, and then to 1 for the other two times. The complementary illegal computation will then be to take the first bit to be equal to 1 the first time, and to 0 the other two times. These two illegal computations will result in encodings that are not at the top level, but their levels will be complementary in the sense that taking their product gives an encoding whose level is twice the top level. We can then use the new zero-test parameter obtained above to determine whether this product of illegal encodings is an encoding of zero or not. It then remains to find a pair of equivalent branching programs such that the illegal encoding obtained above is an encoding of zero for one of the two branching programs only. We exhibit such a pair of branching programs in Section 6.3.4. While we just exhibit one pair, it should be possible to find many other pairs that can also be distinguished. We do not pursue this, as finding one such pair suffices to violate the iO property.

[5] To be correct, we cannot really test whether we have an encoding of 0, but rather whether we have an encoding which is a product of two encodings of 0. More details can be found in Section 6.3.

All the branching program obfuscators mentioned above have a similar structure. In order to simplify the description of the attack, and to highlight which characteristics of these obfuscators are needed for the attack, we first describe in Section 6.2 an abstract obfuscator which captures the obfuscators of [PST14, AGIS14, MSW14, GMM$^+$16]. This abstract obfuscator is elementary, and it suffices to describe our attack against it, in order to attack all the obfuscators of [PST14, AGIS14, MSW14, GMM$^+$16]. The obfuscator of [BGK$^+$14] does not completely fit in this abstract obfuscator and is discussed later.

Discussion and extension. We observe that the mixed-input attack described below crucially relies on the use of straddling set systems. This may seem paradoxical, as straddling set systems were introduced to build obfuscators secure in idealized models, hence supposedly more secure than the first candidates. The first candidate obfuscators [GGH$^+$13b, BR14] tried to prevent mixed-input attacks by using so-called bundling scalars, but it was heuristic and came with no proof. On the contrary, the use of straddling set systems allows us to *prove* that the schemes are resistant to mixed-input attacks if the underlying multilinear map is somehow ideal, hence giving us a security proof in some idealized model. However, this comes at the cost of relying more on the security of the underlying multilinear map. So when the obfuscators are instantiated with the GGH13 multilinear map, which is known to have some weaknesses, this gives more possibilities to an attacker to transform these weaknesses of the multilinear map into weaknesses of the obfuscators. This is what we do here, by transforming a weakness of the GGH13 map into a concrete attack against obfuscators using straddling set systems. It also explains why our attack does not apply to the obfuscators of [GGH$^+$13b, BR14], which did not use straddling set systems.

The mixed-input attack described below can be adapted to quantumly attack all the candidate circuits obfuscators [AB15, Zim15, BD16, DGG$^+$18], when instantiated with the GGH13 multilinear map. While the techniques of the attack against the circuit obfuscators are very similar to the one presented below (they also use straddling set systems), the description of the circuit obfuscators is rather cumbersome. For this reason, the extension to circuit obfuscators is postponed to Appendix B.

6.2 An abstract matrix branching program obfuscator

Following an idea of Miles, Sahai and Zhandry in [MSZ16], we define here an abstract obfuscation scheme. This abstract obfuscator is inspired by the one of [MSZ16] but is a little simpler and more general. In particular, it captures all the obfuscators of Theorem 6.5, except the one of [BGK$^+$14]. We will then show in Section 6.3 how to apply our quantum attack to this abstract obfuscator, resulting in an attack against the obfuscators of [PST14, AGIS14, MSW14, GMM$^+$16] and we will explain how to adapt the attack to the branching program obfuscator of [BGK$^+$14] (which is just slightly different from the abstract obfuscator defined in this section).

The abstract obfuscator takes as input a polynomial size d-ary matrix branching program $\mathbf{A}$ of width w and length ℓ (for some fixed integers $d, w, \ell > 0$), over the ring of integers $\mathbb{Z}$,[6] with a fixed input function $\mathtt{inp}$ and whose matrices have coefficients in $\{0, 1\}$. Typically, the obfuscators pad the branching program with identity matrices, to ensure that the input function has the desired structure. Here, to simplify the obfuscator, we will assume that the obfuscator only accepts branching programs with the desired $\mathtt{inp}$ function (the user has to pad the branching program itself before giving it to the obfuscator). For the attack to work, we ask that there exist two different integers j_1 and j_2 such that $\mathtt{inp}(j_1) \cap \mathtt{inp}(j_2) \neq \emptyset$ (meaning that there is a bit of the input which is inspected at least twice during the evaluation of the branching program). This can be assumed for all the obfuscators of Theorem 6.5.[7] Let A_0, $A_{\ell+1}$ be the bookend vectors of $\mathbf{A}$ and $\{A_{i,\mathbf{b}}\}_{i\in[\ell],\mathbf{b}\in\{0,1\}^d}$ be its square matrices. Recall that the function computed by the branching program $\mathbf{A}$ is defined by

$$x \mapsto \mathbf{A}(x) = \begin{cases} 0 & \text{if } A_0 \cdot \left(\prod_{i\in[\ell]} A_{i,x[\mathtt{inp}(i)]}\right) \cdot A_{\ell+1} = 0 \\ 1 & \text{otherwise.} \end{cases}$$

The abstract obfuscator then proceeds as follows.

[6] Most of the time, the matrices of the branching program will be permutation matrices, and the underlying ring will have no importance.

[7] This is even mandatory for the dual input versions of the obfuscators, as they require that all pairs (s, t) (or (t, s)) appear in the $\mathtt{inp}$ function, for any $s, t \in [m]$ with $s \neq t$.

- It instantiates the GGH13 multilinear map and retains its secret key $(g, h, \{z_i\}_{i\in[\kappa]})$ and its public key (n, q, κ, p_{zt}). The choice of the parameters of the GGH13 map depends on the parameters ℓ, w and d of the branching program $\mathbf{A}$.

- It transforms the matrices of branching program $\mathbf{A}$ to obtain a new branching program $\hat{\mathbf{A}}$, with the same parameters d, ℓ, the same input function $\mathtt{inp}$, a potentially different width $\hat{w}$ and which is strongly equivalent to $\mathbf{A}$. We denote by $\{\hat{A}_{i,\mathbf{b}}\}_{i\in[\ell],\mathbf{b}\in\{0,1\}^d} \in (R/gR)^{\hat{w}\times\hat{w}}$ and $\hat{A}_0 \in (R/gR)^{1\times\hat{w}}, \hat{A}_{\ell+1} \in (R/gR)^{\hat{w}\times 1}$ the matrices and bookend vectors of $\hat{\mathbf{A}}$. Note that this new matrix branching programs has its coefficients in the ring R/gR and not in $\{0,1\}$. Recall that strong equivalence means that

$$A_0 \cdot \prod_{i\in[\ell]} A_{i,\mathbf{b}_i} \cdot A_{\ell+1} = 0 \iff \hat{A}_0 \cdot \prod_{i\in[\ell]} \hat{A}_{i,\mathbf{b}_i} \cdot \hat{A}_{\ell+1} = 0 \text{ (in } R/gR) \tag{6.1}$$

  for all choices of $\mathbf{b}_i \in \{0,1\}^d$, with $i \in [\ell]$. This condition is required for our attack to work, and is satisfied by all the obfuscators of [PST14, AGIS14, MSW14, GMM$^+$16]. To transform the initial branching program $\mathbf{A}$ into this new branching program $\hat{\mathbf{A}}$, the obfuscators of [PST14, AGIS14, MSW14, GMM$^+$16] first embed the matrices of $\mathbf{A}$ into the ring R/gR (this is possible since the coefficients of the matrices are binary). Then, they use various tools, taken among the following.[8]

  1. Transform the matrices $A_{i,\mathbf{b}}$ into block-diagonal matrices $\begin{pmatrix} A_{i,\mathbf{b}} & \\ & B_{i,\mathbf{b}} \end{pmatrix}$, were $B_{i,\mathbf{b}}$ are square $w' \times w'$ matrices in R/gR, chosen arbitrarily (they can be fixed, or chosen at random, this will have no importance for us), with w' polynomial in the security parameter λ. In order to cancel the extra diagonal block, the vector A_0 is transformed into $\begin{pmatrix} A_0 & 0 \end{pmatrix}$, with a block of zeros of size $1 \times w'$. The vector $A_{\ell+1}$ is transformed into $\begin{pmatrix} A_{\ell+1} \\ B_{\ell+1} \end{pmatrix}$, with $B_{\ell+1}$ an arbitrary $w' \times 1$ vector.
  2. Use Killian randomisation, that is, choose $\ell+1$ non singular matrices $\{R_i\}_{i\in[\ell+1]} \in (R/gR)^{w\times w}$ and transform $A_{i,\mathbf{b}}$ into $R_i \cdot A_{i,\mathbf{b}} \cdot R_{i+1}^{\text{adj}}$, where R_{i+1}^{adj} is the adjugate matrix of R_{i+1}, i.e., $R_{i+1} \cdot R_{i+1}^{\text{adj}} = \det(R_{i+1}) \cdot I_w$. Transform also A_0 into $A_0 \cdot R_1^{\text{adj}}$ and $A_{\ell+1}$ into $R_{\ell+1} \cdot A_{\ell+1}$.
  3. Multiply by random scalars, i.e., multiply each matrix $A_{i,\mathbf{b}}$ by some random scalar $\alpha_{i,\mathbf{b}} \in (R/gR)^\times$. Also multiply A_0 and $A_{\ell+1}$ by α_0 and $\alpha_{\ell+1}$ respectively.

  One can check that all the transformations described above output a branching program which is strongly equivalent to the one given as input, so the final branching program $\hat{\mathbf{A}}$ is also strongly equivalent to $\mathbf{A}$ (as in (6.1)). In the following, we will only be interested in (6.1), not in the details of the transformation.

- Finally, the obfuscator encodes the matrices $\{\hat{A}_{i,\mathbf{b}}\}_{i,\mathbf{b}}$, $\hat{A}_0$ and $\hat{A}_{\ell+1}$ coefficient-wise, at some levels $\{\boldsymbol{v}_{i,\mathbf{b}}\}_{i,\mathbf{b}}$, $\boldsymbol{v}_0$ and $\boldsymbol{v}_{\ell+1}$ respectively, using the GGH13 multilinear map. The choice of these levels (called a straddling set system) depends on the obfuscators, but will have no importance in the following. The only property that we need, and that is fulfilled by the above obfuscators, is that for any entry x, the sets $\boldsymbol{v}_0, \boldsymbol{v}_{\ell+1}$ and $\boldsymbol{v}_{i,x[\mathtt{inp}(i)]}$ for $i \in [l]$ are disjoint and we have

$$\boldsymbol{v}_0 + \left(\sum_{i\in[l]} \boldsymbol{v}_{i,x[\mathtt{inp}(i)]} \right) + \boldsymbol{v}_{\ell+1} = \boldsymbol{v}^*. \tag{6.2}$$

  Recall that $\boldsymbol{v}^*$ is the maximum level of the GGH13 map. This means that every honest evaluation of the encoded branching program outputs an element at level $\boldsymbol{v}^*$, which can be zero-tested. This condition is necessary for the above obfuscators to be correct (otherwise we cannot evaluate the obfuscated branching program). In the following, we will write $[A]_{\boldsymbol{v}}$ the matrix (or vector) obtained by encoding each coefficient of A individually at level $\boldsymbol{v}$, using the GGH13 map. We will call such a matrix (or vector) an encoding of A at level $\boldsymbol{v}$.

[8] The obfuscators of [PST14, GMM$^+$16] use the three tools while the ones of [AGIS14, MSW14] use Tools 2 and 3 only.

- The obfuscator then outputs the elements $[\hat{A}_0]_{\boldsymbol{v}_0}$, $\{[\hat{A}_{i,\mathbf{b}}]_{\boldsymbol{v}_{i,\mathbf{b}}}\}_{i\in[l],\mathbf{b}\in\{0,1\}^d}$, $[\hat{A}_{\ell+1}]_{\boldsymbol{v}_{\ell+1}}$ and the public parameters of the GGH13 map (n, q, κ, p_{zt}).

To evaluate the obfuscated branching program on input x, compute

$$u_x = [\hat{A}_0]_{\boldsymbol{v}_0} \times \prod_{i\in[\ell]} [\hat{A}_{i,x[\mathtt{inp}(i)]}]_{\boldsymbol{v}_{i,x[\mathtt{inp}(i)]}} \times [\hat{A}_{\ell+1}]_{\boldsymbol{v}_{\ell+1}}.$$

By Property (6.1), this is an encoding of zero if and only if the output of the original branching program was zero, and by Property (6.2), this encoding is at level $\boldsymbol{v}^*$. So using p_{zt}, we can perform a zero-test and output 0 if this is an encoding of 0 and 1 otherwise. In the following, we will sometimes simplify notations and forget about the subscripts $\boldsymbol{v}_{i,\mathbf{b}}$, as the levels of the encodings are entirely determined by the encoded matrices $A_{i,\mathbf{b}}$.

6.2.1 Heuristic assumption

For our attack to work, we will need to assume that if we evaluate the obfuscated branching program on sufficiently many inputs for which the output is zero, then we can recover a basis of the ideal $\langle h\rangle$ (where h is a secret element of the GGH13 map, as described in Section 5.2). More formally, we make the following heuristic assumption.

Heuristic 6.6. *Let X_0 be the set of inputs on which the branching program evaluates to 0. If we evaluate the obfuscated branching program on an input $x \in X_0$ and zero-test the final encoding, we obtain a ring element of the form $r_x \cdot h \in R$. We assume that if we choose polynomially many x's independently and uniformly at random in X_0, then, with overwhelming probability, the obtained $r_x \cdot h$ span the ideal $\langle h\rangle$ (and not a smaller ideal contained in $\langle h\rangle$).*

Experimental results. In order to strengthen our confidence in Heuristic 6.6, we ran some computational experiments in Sage, which were consistent with the assumption. We let n be the dimension of the cyclotomic ring R, ℓ be the number of matrices in the branching programs and m be the dimension of the matrices of the branching programs. In our experiments, we only considered branching programs that consist of identity matrices and we took as bookend vectors the vectors $A_0 = \begin{pmatrix}1 & 0 & \cdots & 0\end{pmatrix}$ and $A_{\ell+1} = \begin{pmatrix}0 & \cdots & 0 & 1\end{pmatrix}^T$. We then randomized these branching programs by multiplying them by random bundling scalars and by the random matrices of Killian randomisation (this corresponds to the transformations 2 and 3 of the abstract obfuscator defined in Section 6.2). Finally, the coefficients of these matrices were reduced modulo g using Babai's round-off algorithm. We then evaluated this branching program on a random input (as we took only identity matrices, any combination of these matrices will result in a zero) and divided the resulting element by g. The element obtained this way is called a post-zero-test element. For a fixed g and randomized branching program, we computed many different post-zero-test elements and checked whether they generated the whole ring or not (note that our construction corresponds to the case where we took $h = 1$, hence $\langle h\rangle = R$). For a fixed g, we then computed different randomized branching programs (the underlying branching program always consists of identity matrices, but the bundling scalars and the random matrices are different), and computed the empirical probability, over these randomized branching programs, that the post-zero-test values generate the full ring R.

We then computed this probability of success for many g's, for two sets of parameters:

$$n = 10, \ell = 10 \text{ and } m = 5$$
$$\text{or } n = 32, \ell = 20 \text{ and } m = 5.$$

The elements g are randomly chosen (using the `random_element` procedure of Sage), with rejection to ensure that their algebraic norm is a prime number. The results of these experiments are shown in Tables 6.2 and 6.3. The value 'number of BP' represents the number of different randomizations of branching programs we used to compute the empirical mean and the value 'number of post-zero-test values' represents the number of post-zero-test values we considered to check whether the resulting ideal was equal to $\langle h\rangle$ or not.

Empirical probability to recover $\langle h \rangle$	$[0, 0.1)$	$[0.1, 0.6)$	$[0.6, 0.7)$	$[0.7, 0.8)$	$[0.8, 0.9)$	$[0.9, 1]$
Number of g's	2	0	1	0	9	28

Table 6.2: Experimental results obtained for $n = 10$, $\ell = 10$, $m = 5$, number of BP $= 100$, number of post-zero-test values $= 30$ and total number of $g = 40$.

Empirical probability to recover $\langle h \rangle$	< 0.8	0.8	0.85	0.90	0.95	1
Number of g's	0	1	2	5	7	5

Table 6.3: Experimental results obtained for $n = 32$, $\ell = 20$, $m = 5$, number of BP $= 20$, number of post-zero-test values $= 30$ and total number of $g = 20$.

We observe that for the first choice of parameters, there are some values of g for which we almost never recover a basis of $\langle h \rangle$. This corresponds to choices of g with small algebraic norm (for instance 2 or 13). These exceptional cases disappear when n increases (for the second choice of parameters). As the parameter n should be larger than the security parameter (see Section 5.2.2), these exceptional cases should not impact our attack. Note that in the case where $n = 32$, for every tested g, the probability to recover the ideal $\langle h \rangle$ is empirically at least 0.8. Moreover, we observed that in the cases where we did not recover the ideal $\langle h \rangle$, we recovered a very small multiple of it.

Handling more general cases. Let us also observe that, even if we do not recover the ideal $\langle h \rangle$ exactly, we can still create our new zero-testing parameter in some cases. Let $J \subseteq \langle h \rangle$ be the ideal obtained by zero-testing many encodings of zero. We can factor J into prime ideals in quantum polynomial time, and we know that $\langle h \rangle$ is a product of some of these prime ideals. Assume now that J has a polynomial number of prime factors and that $\langle h \rangle$ has a constant number of prime factors (there is a non negligible probability that this is the case). We can then guess $\langle h \rangle$, by trying exhaustively all possible products of a constant number of prime factors of J. Once we have a candidate ideal J' for $\langle h \rangle$, we apply Theorem 2.21 to recover a generator h' of the ideal. If the algorithm fails, for example because the ideal is not principal or does not have a short generator, we know that $J' \neq \langle h \rangle$. Let then $x \in X_0$ (using the notations of Heuristic 6.6) and $r_x \cdot h$ be the ring element obtained after zero-testing the evaluation of the obfuscated branching program on x. If $r_x h/h'$ is not in R for one of the tested x's, we know that h' is not the good element. We also know that $\|r_x\| \leq q^{1/4}$, because $\|r_x \cdot h\|$ should be less than $q^{3/4}$ and $\|h\|$ is of the order of $q^{1/2}$. Hence, if $\|r_x \cdot h/h'\| > q^{1/4}$, we know that $h' \neq h$ and we try another ideal. On the contrary, if $\|r_x \cdot h/h'\| \leq q^{1/4}$, we may have $h' = h$, but we have no guarantee that this is indeed the case. To improve our confidence, we perform the test above with many inputs $x \in X_0$. Looking forward, even if we recover an element h' which is different from h, as long as $r_x \cdot h/h' \in R$ and $\|r_x \cdot h/h'\| \leq q^{1/4}$ with good probability over the choice of $x \in X_0$ (which will be the case if h' passes the tests above for many random inputs x), then we will be able to perform our attack using h' instead of h. To sum up, if we recover an ideal $J \neq \langle h \rangle$ which has a polynomial number of prime factors and if $\langle h \rangle$ has a constant number of prime factors, then we can still perform our attack, even if Heuristic 6.6 does not hold.

This completes the definition of our abstract obfuscator, which captures the obfuscators of [PST14, AGIS14,MSW14,GMM$^+$16]. In the next section, we describe a mixed-input attack against this abstract obfuscator, where all we use is that it satisfies Properties (6.1) and (6.2).

6.3 Quantum attack against the abstract obfuscator

We will now prove our main theorem.

Theorem 6.7. *Let $\mathcal{O}$ be any of the obfuscators in [BGK$^+$14, PST14, AGIS14, MSW14, GMM$^+$16], on single or dual input branching programs, instantiated with the GGH13 multilinear map [GGH13a]. Assume the secret parameter h of the GGH13 multilinear map is sampled using a spherical Gaussian distribution. Then, there exist two explicit equivalent branching programs $\boldsymbol{A}$ and $\boldsymbol{A}'$ such that $\mathcal{O}(\boldsymbol{A})$ and $\mathcal{O}(\boldsymbol{A}')$ can be distinguished in quantum polynomial time, under GRH, Conjecture 2.22 and Heuristic 6.6.*

The necessity for h being sampled according to a spherical Gaussian distribution appears in Theorem 2.21, to solve the short Principal Ideal Problem and recover the secret element h. It is not used anywhere else in the attack, in particular, it is not used in the mixed-input part of the attack (see Section 6.3.3). We discuss this limitation to spherical Gaussian distribution in Section 6.3.2. We will see that the general case is related to the problem of finding small elements in a given ideal, which we discussed in Chapter 3.

To prove Theorem 6.7, we present a quantum polynomial time distinguishing attack against the abstract obfuscator described in Section 6.2. This results into an attack against the iO security of the branching program obfuscators of [PST14, AGIS14, MSW14, GMM+16]. We then explain how to slightly modify this attack to use it against the obfuscator of [BGK+14], whose structure is very close to the one of the abstract obfuscator.

The attack works in two steps. We first recover the secret element h of the GGH13 multilinear map. Using the results of [BS16, CGS14a, CDPR16], recalled in Section 2.3.10, this can be done in quantum polynomial time. Knowing this secret element h, we are able to construct a zero-testing parameter p'_{zt} at a higher level than $\boldsymbol{v}^*$. We can then use this new parameter p'_{zt} to mount a (classical) polynomial time mixed-input attack against the abstract obfuscator.

6.3.1 Creating a new zero-testing parameter

We first explain in this section how we can recover the secret parameter h of the multilinear map in quantum polynomial time. We then describe how to construct a new zero-testing parameter at a level higher than $\boldsymbol{v}^*$, using h. Note that the following is folklore, we recall it for the sake of completeness.

The first step is to recover sufficiently many multiples of h, to obtain a basis of the ideal $\langle h \rangle$ (when seen as a sub-lattice of R). This part of the attack was already described in the original article [GGH13a], and can be done in classical polynomial time, under Heuristic 6.6. Observe that for each top-level encoding that passes the zero-test, we obtain a multiple of h. We rely on Heuristic 6.6 to ensure that we indeed recover a basis of the ideal $\langle h \rangle$, by zero-testing sufficiently many top-level encodings of zero. For this step to work, we need that the branching program evaluates sufficiently often to 0, to obtain sufficiently many encodings of 0. In the following, we will choose branching programs that compute the always zero function, hence the condition on the number of encodings that pass the zero-test will be satisfied.

We then recover $\pm X^i h$ from a basis of the ideal $\langle h \rangle$ in polynomial quantum time, under Conjecture 2.22 and GRH (see Theorem 2.21). Observe that we can indeed apply the theorem as h is sampled according to a spherical Gaussian distribution of parameter larger than $200 \cdot n^{1.5}$. The fact that we recover $\pm X^j h$ instead of h will have no importance for our attack,[9] so in the following we will assume that we recovered h exactly.

We now explain how to use h to create a new zero-testing parameter p'_{zt} at a higher level than $\boldsymbol{v}^*$. A close variant of this step was already mentioned in [GGH13a, Section 6.3.3]. The authors explained how to use a small multiple of $1/h$ and a low level encoding of zero to create a new zero-testing parameter that enables them to test at a higher level whether the numerator of an encoding is a multiple of g or not (i.e., if the encoding is an encoding of zero or not). In our case, the situation is a little different, as we do not know any low level encoding of zero. Hence, we only manage to create a new zero-testing parameter that enables us to determine whether the numerator of an encoding is a multiple of g^2 or not, at level $2\boldsymbol{v}^*$. Remember that this is an invalid level, but we can still create encodings at this level (for instance by multiplying two level $\boldsymbol{v}^*$ encodings). We use the secret h to compute a new zero-testing parameter p'_{zt} at level $2\boldsymbol{v}^*$. Recall that $p_{zt} = hz^*g^{-1} \bmod q$. We then define

$$p'_{zt} = p_{zt}^2 h^{-2} \bmod q = (z^*)^2 \cdot g^{-2} \bmod q.$$

Again, note that even if we call it a new zero-testing parameter, p'_{zt} only enables us to test whether the numerator of a level $2\boldsymbol{v}^*$ encoding is a multiple of g^2, and not g, as our original zero-test parameter p_{zt} did. Still, being able to test at a level higher than $\boldsymbol{v}^*$ if the numerator is a multiple of g^2 will enable us to mount a mixed-input attack against the abstract obfuscator of Section 6.2. We describe this mixed-input attack in the next subsection.

[9] This is because both X^j and its inverse $-X^{n-j}$ have euclidean norm 1.

6.3.2 Non-spherical Gaussian distributions

Recall that to use Theorem 2.21 to recover the element h from the ideal $\langle h \rangle$, we have to assume that h is sampled according to a spherical Gaussian distribution. In this section, we explain how being able to solve approx-SVP in ideal lattices can help us create a new zero-testing parameter for any distribution of h. This shows that being able to find short vectors in ideal lattices indeed has some impact on the security of the GGH13 map.

First, let us observe that in order to create our new zero-testing parameter p'_{zt}, it is not necessary to recover h exactly. Indeed, assume we can recover a small multiple of $1/h$, namely an element w/h where $w \in R$ and $\|w\| \leq q^{1/4-\varepsilon}$ for some $\varepsilon > 0$. Then, we can define $p'_{zt} = p_{zt}^2 \cdot (w/h)^2 = w^2 \cdot (z^*)^2 \cdot g^{-2} \bmod q$. Now, recall that a top-level encoding of zero $rg/z^* \bmod q$ should satisfy $\|r\| \leq q^{1/4}$ because we want $\|rh\| \leq q^{3/4}$ and $\|h\|$ is of the order of $\sqrt{q}$. Hence, if we multiply two top-level encodings of zero $u_1 = r_1 g/z^* \bmod q$ and $u_2 = r_2 g/z^* \bmod q$ by our new zero-testing parameter, we obtain

$$u_1 \cdot u_2 \cdot p'_{zt} = r_1 \cdot r_2 \cdot w^2 \bmod q.$$

Thanks to our upper bound on $\|r_1\|, \|r_2\|$ and $\|w\|$ we conclude that $\|r_1 \cdot r_2 \cdot w^2\| \leq q^{1-2\varepsilon} \ll q$. Hence, we can zero-test at level $2\boldsymbol{v}^*$ by multiplying an encoding at level $2\boldsymbol{v}^*$ by p'_{zt} and checking whether the resulting product is smaller than $q^{1-2\varepsilon}$ or not.

Our objective is then to recover a small multiple of $1/h$ given a basis of the ideal $\langle h \rangle$ and some small multiples $r_i h$ of h, obtained by zero-testing some top-level encodings of zero. This problem is related to the approx-SVP problem in ideal lattices (see Chapter 3).

We will decompose our algorithm to recover a small multiple of $1/h$ into two steps. The first one consists in finding an element $\alpha \in K_{\mathbb{R}}$ which is not too far from h. In this chapter, because the elements we consider are in coefficient embeddings, we view $K_{\mathbb{R}}$ as $\mathbb{R}[X]/(X^n + 1)$. This element α has no algebraic relation with h, the condition is purely geometric. More formally, assume we have computed polynomially many multiples $r_i h$ of h, by zero-testing top-level encodings of zero. We would like that for all i, the elements $r_i h/\alpha$ have small infinity norm (significantly smaller than $q^{1/2}$). Then, if we could divide p_{zt}^2 by α^2, we would obtain our zero-test at level $2\boldsymbol{v}^*$ (because a product of two such $r_i h/\alpha$ would have infinity norm significantly smaller than q). However, we cannot divide p_{zt} by α, because this is not a multiple of h. Hence, if we try to divide the zero-testing parameter by α modulo q, we would just obtain $r_i h/\alpha \bmod q$, where the division is performed modulo q and might be large (and is not even well-defined when $\alpha \notin K$). The second step of the algorithm is then to obtain a multiple of $1/h$ relatively close to $1/\alpha$, and use it to create our new zero-testing parameter.

Step 1. In order to recover an element $\alpha \in K_{\mathbb{R}}$ close to h, we assume that we are given a polynomial number of $r_i h$, obtained by zero-testing top-level encodings of zero. We know that $\|r_i\|_\infty \leq q^{1/4}$ for all i, hence, taking the logarithm we obtain that $\mathrm{Log}(r_i h)_j \leq \mathrm{Log}(h)_j + \frac{1}{4}\log q + O(\log n)$ for all j-th coordinates, with $1 \leq j \leq n$ (the $O(\log n)$ term comes from the transformation between coefficient embeddings and canonical embeddings). Let us assume in the following that $\log n$ is negligible compared to $\log q$. We solve the following optimization problem over $\mathbb{R}$ (see Figure 6.5 for an example in dimension 2).

$$\begin{aligned} \text{Minimize} \quad & \sum_j x_j \\ \text{Subject to} \quad & \mathrm{Log}(r_i h)_j \leq x_j + \frac{1}{4}\log q \quad \text{for all } i, j. \end{aligned}$$

This optimization problem can be solved in polynomial time (with $x_j \in \mathbb{R}$) using linear programming. Recall that the Log of an element $\alpha \in \mathbb{R}[X]/(X^n+1)$ is defined as $\mathrm{Log}(\alpha) = (\log|\alpha(\zeta_1)|, \ldots, \log|\alpha(\zeta_n)|)$, where $\zeta_1, \ldots, \zeta_n$ are the roots of X^n+1 ordered such that $\zeta_i = \overline{\zeta_{i+n/2}}$ for all $1 \leq i \leq n/2$. Now, observe that our optimization problem above has symmetries (we have $\mathrm{Log}(r_i h)_j = \mathrm{Log}(r_i h)_{j+n/2}$ for all i and $1 \leq j \leq n/2$), hence we can find a solution x such that $x_j = x_{j+n/2}$ for all $1 \leq j \leq n/2$ (given any solution, we can take its symmetric because the constraints of the problem are symmetric, and then take the middle point of the two solutions because the set of solutions is convex).

Using this remark about the symmetries of x, we will be able to compute $\alpha \in K_{\mathbb{R}} = \mathbb{R}[X]/(X^n+1)$ such that $\mathrm{Log}\,\alpha = x$, by polynomial interpolation. More precisely, we are looking for a polynomial

$\alpha \in \mathbb{R}[X]$ of degree at most $n-1$ such that $|\alpha(\zeta_i)| = \mathrm{e}^{x_i}$ for all i's. Observe that we only have a constraint on the modules of the $\alpha(\zeta_i)$, and so we can arbitrarily set their arguments. However, if we want to find α with real coefficients, we have to choose the arguments such that the evaluations of α at conjugate roots give conjugate evaluations. A simple choice satisfying these constraints is to choose all the arguments to be 0 (we already know that the modules of the evaluation at conjugate roots are the same thanks to the previous remark), and look for α such that $\alpha(\zeta_i) = \mathrm{e}^{x_i}$. By polynomial interpolation, we can find in polynomial time the unique $\alpha \in \mathbb{C}[X]$ of degree $< n$ such that $\alpha(\zeta_i) = \mathrm{e}^{x_i}$ for all $1 \leq i \leq n$. Let us conclude by proving that α has real coefficients. Because the e^{x_i}'s are real and the roots of $X^n + 1$ are closed under conjugation, we know that

$$\overline{\alpha}(\zeta_i) = \overline{\alpha(\zeta_{i+n/2})} = \overline{\mathrm{e}^{x_{i+n/2}}} = \mathrm{e}^{x_{i+n/2}} = \mathrm{e}^{x_i} = \alpha(\zeta_i).$$

This means that α and $\overline{\alpha}$ coincide at n distinct points, and so should be equal. Hence, we conclude that $\alpha \in \mathbb{R}[X]$ and satisfies $\mathrm{Log}\,\alpha = x$.

This element α is a good candidate for h. Indeed, because $\mathrm{Log}(r_i h)_j \leq \mathrm{Log}(\alpha)_j + \frac{1}{4}\log q$ for all i, j, then we have that

$$\|\mathrm{Log}(\frac{r_i h}{\alpha})\|_\infty \leq \frac{1}{4}\log q,$$

which can be rewritten as $\|r_i h/\alpha\|_\infty \leq q^{1/4}$ for all i. Observe that this does not necessarily imply that $\|rh/\alpha\|_\infty \leq q^{1/4}$ for all zero-tested top-level encodings of zero. However, if we randomly chose the top-level encodings of zero used to create the $r_i h$, then, by ensuring that $\|r_i h/\alpha\|_\infty \leq q^{1/4}$ for all i, we expect that with good probability we will also have $\|rh/\alpha\|_\infty \leq q^{1/4}$ for a new top-level encoding of zero.

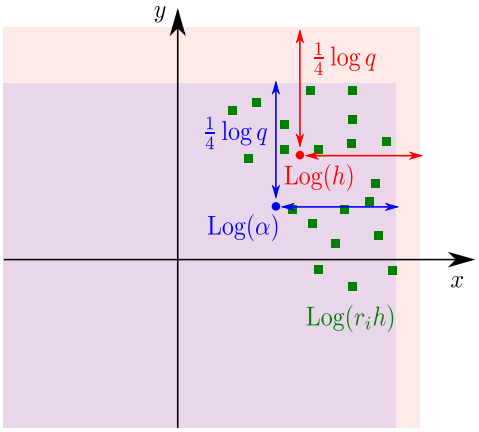

Figure 6.5: Illustration of the choice of α

Step 2. Observe that h satisfies the linear constraints of our optimization problem, hence, by choice of α, we know that $\sum_j \mathrm{Log}(\alpha)_j \leq \sum_j \mathrm{Log}(h)_j$. This implies that $|\mathcal{N}(\alpha)| \leq |\mathcal{N}(h)|$. We now solve approx-SVP in $\alpha \cdot \langle \frac{1}{h} \rangle$, for some approximation factor γ, and obtain a solution $x = \frac{\alpha w}{h}$, with $w \in R$. We know that $\|x\| \leq \gamma \cdot \sqrt{n} \cdot |\mathcal{N}(\alpha)|^{1/d} \cdot |\mathcal{N}(h)|^{-1/d} \leq \gamma \cdot \sqrt{n}$.

Let us now show that w/h can be used to create our level-$2\boldsymbol{v}^*$ zero-testing parameter, by setting $p'_{zt} = p_{zt}^2 \cdot (w/h)^2 \bmod q = (z^* w/g)^2 \bmod q$. When we multiply it by two top-level encodings of zero $u_1 = r_1 g/z^*$ and $u_2 = r_2 g/z^*$, we obtain $r_1 r_2 w^2 \bmod q$, which we can recover exactly if $\|r_1 r_2 w^2\|_\infty < q$ (because r_1, r_2 and w are in R). By definition of w, we know that for $i = 1, 2$ we have

$$\begin{aligned}
\|r_i w\|_\infty &= \|\frac{r_i h}{\alpha} \cdot x\|_\infty \\
&\leq \|\frac{r_i h}{\alpha}\|_\infty \cdot \|x\|_1 \qquad \text{(see Equation (2.8))} \\
&\leq q^{1/4} \cdot n \cdot \gamma,
\end{aligned}$$

where we used the fact that α is chosen so that $\|\frac{r_i h}{\alpha}\|_\infty \leq q^{1/4}$ with good probability. Taking the product, we obtain that $\|r_1 r_2 w^2\|_\infty \leq q^{1/2} \cdot n^3 \cdot \gamma^2$. Hence, as long as $\gamma \leq \frac{q^{1/4-\varepsilon}}{n^{1.5}}$ for some $\varepsilon > 0$, then we have $\|r_1 r_2 w^2\|_\infty < q^{1-2\varepsilon} \ll q$, and we can test whether a level-$2\boldsymbol{v}^*$ encoding is a product of two encodings of zero.

We conclude that if we can solve approx-SVP in principal ideal lattices (multiplied by an element $\alpha \in K_\mathbb{R}$) with approximation factor $\gamma \leq \frac{q^{1/4-\varepsilon}}{n^{1.5}}$ for some $\varepsilon > 0$, then we can create a zero-testing parameter p'_{zt} at level $2\boldsymbol{v}^*$ for any distribution of the parameter h.

6.3.3 The mixed-input attack

We now assume that we have built a new pseudo-zero-test parameter p'_{zt}, as in Section 6.3.1 (in quantum polynomial time), and that we are given an obfuscated branching program $([\hat{A}_0]_{\boldsymbol{v}_0}, \{[\hat{A}_{i,\mathbf{b}}]_{\boldsymbol{v}_{i,\mathbf{b}}}\}_{i\in[l],\mathbf{b}\in\{0,1\}^d}, [\hat{A}_{\ell+1}]_{\boldsymbol{v}_{\ell+1}})$, obtained by using our abstract obfuscator defined in Section 6.2.

Let x and y be two different inputs of the branching program. A mixed-input attack consists in changing the value of some bits of the input during the evaluation of the obfuscated branching program. For instance, the way we will do it is by taking some matrix $[\hat{A}_{i,y[\mathtt{inp}(i)]}]_{\boldsymbol{v}_{i,y[\mathtt{inp}(i)]}}$, instead of $[\hat{A}_{i,x[\mathtt{inp}(i)]}]_{\boldsymbol{v}_{i,x[\mathtt{inp}(i)]}}$, while evaluating the program on x. Such mixed-input attack can leak information on the program being obfuscated (see the specific choice of branching programs described in the next subsection). In order to prevent mixed-input attack, the abstract obfuscator uses a straddling set system. The intuition is that if the attacker tries to mix the matrices $[\hat{A}_{i,x[\mathtt{inp}(i)]}]_{\boldsymbol{v}_{i,x[\mathtt{inp}(i)]}}$ and $[\hat{A}_{i,y[\mathtt{inp}(i)]}]_{\boldsymbol{v}_{i,y[\mathtt{inp}(i)]}}$, it will not get an encoding at level $\boldsymbol{v}^*$ at the end of the computation and hence it cannot zero-test it. However, we can use our new zero-testing parameter p'_{zt} to circumvent this difficulty.

Let $j \in [\ell]$ and compute

$$\tilde{u}_{x,j} = [\hat{A}_0] \cdot \left(\prod_{i<j} [\hat{A}_{i,x[\mathtt{inp}(i)]}]\right) \cdot [\hat{A}_{j,y[\mathtt{inp}(j)]}] \cdot \left(\prod_{j<i\leq\ell} [\hat{A}_{i,x[\mathtt{inp}(i)]}]\right) \cdot [\hat{A}_{\ell+1}]$$

$$\tilde{u}_{y,j} = [\hat{A}_0] \cdot \left(\prod_{i<j} [\hat{A}_{i,y[\mathtt{inp}(i)]}]\right) \cdot [\hat{A}_{j,x[\mathtt{inp}(j)]}] \cdot \left(\prod_{j<i\leq\ell} [\hat{A}_{i,y[\mathtt{inp}(i)]}]\right) \cdot [\hat{A}_{\ell+1}],$$

that is, we exchange $[\hat{A}_{j,x[\mathtt{inp}(j)]}]_{\boldsymbol{v}_{j,x[\mathtt{inp}(j)]}}$ and $[\hat{A}_{j,y[\mathtt{inp}(j)]}]_{\boldsymbol{v}_{j,y[\mathtt{inp}(j)]}}$ in the honest evaluations of the obfuscated branching program on inputs x and y.

The encodings $\tilde{u}_{x,j}$ and $\tilde{u}_{y,j}$ will have illegal levels $\boldsymbol{v}_x$ and $\boldsymbol{v}_y$ that are different from $\boldsymbol{v}^*$. However, as we only exchange two matrices between correct evaluations, we know that $\tilde{u}_{x,j} \cdot \tilde{u}_{y,j}$ will be encoded at the same level as $u_x \cdot u_y$ where u_x and u_y are the correct evaluations of the obfuscated branching program on x and y. As u_x and u_y are correct evaluations, using Property (6.2), we know that they are encoded at level $\boldsymbol{v}^*$. Hence $\tilde{u}_{x,j} \cdot \tilde{u}_{y,j}$ is encoded at level $2\boldsymbol{v}^*$, and we can zero-test $\tilde{u}_{x,j} \cdot \tilde{u}_{y,j}$ using p'_{zt}.

Remember that an encoding will pass this zero-test only if its numerator is a multiple of g^2 and not only g. A simple way to ensure that $\tilde{u}_{x,j} \cdot \tilde{u}_{y,j}$ has a numerator which is a multiple of g^2 is to choose x and y such that $\tilde{u}_{x,j}$ and $\tilde{u}_{y,j}$ are both encodings of 0 (i.e., their numerator are both multiples of g, and hence their product has a numerator which is a multiple of g^2). Using Property (6.1) of our abstract obfuscator, we know that $\tilde{u}_{x,j}$ is an encoding of 0 if and only if

$$A_0 \cdot \left(\prod_{i<j} A_{i,x[\mathtt{inp}(i)]}\right) \cdot A_{j,y[\mathtt{inp}(j)]} \cdot \left(\prod_{j<i\leq\ell} A_{i,x[\mathtt{inp}(i)]}\right) \cdot A_{\ell+1} = 0.$$

We let $\tilde{a}_{x,j}$ denote the left hand side of this equation. In the same way, we define

$$\tilde{a}_{y,j} = A_0 \cdot \left(\prod_{i<j} A_{i,y[\mathtt{inp}(i)]}\right) \cdot A_{j,x[\mathtt{inp}(j)]} \cdot \left(\prod_{j<i\leq\ell} A_{i,y[\mathtt{inp}(i)]}\right) \cdot A_{\ell+1},$$

and we know that $\tilde{u}_{y,j}$ is an encoding of 0 if and only if $\tilde{a}_{y,j} = 0$.

To conclude, if we manage to find two equivalent branching programs $\mathbf{A}$ and $\mathbf{A}'$, two inputs x and y and an integer $j \in [\ell]$ such that $\tilde{a}_{x,j} = \tilde{a}_{y,j} = 0$ for $\mathbf{A}$ but $\tilde{a}'_{x,j} \neq 0$ and $\tilde{a}'_{y,j} \neq 0$ for $\mathbf{A}'$, then we can distinguish between the obfuscation of $\mathbf{A}$ and the one of $\mathbf{A}'$. Indeed, the numerator of $\tilde{u}_{x,j} \cdot \tilde{u}_{y,j}$ will be a multiple of g^2 in the case of $\mathbf{A}$ but the numerator of $\tilde{u}'_{x,j} \cdot \tilde{u}'_{y,j}$ will not be a multiple of g in the case of $\mathbf{A}'$ (and therefore not a multiple of g^2 either). Hence, using p'_{zt}, we can determine which of the branching program $\mathbf{A}$ or $\mathbf{A}'$ has been obfuscated.

In the next subsection, we present two possible branching programs $\mathbf{A}$ and $\mathbf{A}'$ and inputs x and y that satisfy the condition above. We note that this condition is easily satisfied and it should be possible to find a lot of other branching programs satisfying it. We just propose here a simple example of such branching programs, in order to complete the proof of Theorem 6.5.

6.3.4 A concrete example of distinguishable branching programs

In this section, we present an example of two branching programs $\mathbf{A}$ and $\mathbf{A}'$ that are equivalent, but such that their obfuscated versions, obtained using the abstract obfuscator, can be distinguished using the framework described above, hence breaking the iO security of the obfuscator.

Remember that for the first step of our attack (recovering h and creating p'_{zt}, see Section 6.3.1), we need to have sufficiently many inputs x which evaluate to zero. Here, we choose branching programs that compute the always zero function. We now show how to satisfy the conditions for the second part of the attack (Section 6.3.3).

Let $I = I_w \in \{0,1\}^{w \times w}$ be the identity matrix and $J \in \{0,1\}^{w \times w}$ be a matrix of order two (i.e., $J \neq I$ and $J^2 = I$). One could for example take

$$J = \begin{pmatrix} 0 & 1 & & \\ 1 & 0 & & \\ & & I_{w-2} & \end{pmatrix}. \tag{6.3}$$

Our first branching program will consist in identity matrices only. We will build our second branching program such that when evaluating it on input x, we have a product of ℓ matrices I (when we forget about the bookend vectors), but on input y we have a product of $\ell - 2$ matrices I and 2 matrices J.[10] We will then exchange one of these J matrices with an I matrix in the evaluation on x. The resulting products will then be equal to the matrix J instead of the matrix I (as it is the case for the first branching program). We describe the two branching programs more precisely below.

Input selection function. Recall that the input selection function $\texttt{inp}$ is fixed and is such that there are at least two distinct integers j_1 and j_2 such that $\texttt{inp}(j_1) \cap \texttt{inp}(j_2) \neq \emptyset$. Let s be such that $s \in \texttt{inp}(j_1) \cap \texttt{inp}(j_2)$. This means that when evaluating the branching program on some input, the j_1-th and the j_2-th matrices of the product both depend on the s-th bit of the input. Without loss of generality, we assume that $\texttt{inp}(j_1) = (s, s_2, \cdots, s_d)$ and $\texttt{inp}(j_2) = (s, t_2, \cdots, t_d)$ for some integers s_i and t_i in $[m]$.

Matrices. Our first branching program $\mathbf{A}$ consists in identity matrices only, i.e., $A_{i,\mathbf{b}} = I$ for all $i \in [\ell]$ and $\mathbf{b} \in \{0,1\}^d$. For our second branching program $\mathbf{A}'$, we take

$$A'_{i,\mathbf{b}} = \begin{cases} I & \text{if } i \notin \{j_1, j_2\} \text{ or } b_1 = 0 \\ J & \text{if } i \in \{j_1, j_2\} \text{ and } b_1 = 1, \end{cases}$$

where $\mathbf{b} = (b_1, \cdots, b_d)$. This means that when evaluating the branching program $\mathbf{A}'$ on some input x, if $x_s = 0$, then all the matrices of the product are identity matrices. On the contrary, if $x_s = 1$, then the j_1-th and j_2-th matrices of the product are J matrices and the others are I matrices. As J has order two, the product will always be the identity.

Bookend vectors. We take A_0 and $A_{\ell+1}$ to be two vectors such that $A_0 I A_{\ell+1} = 0$ but $A_0 J A_{\ell+1} \neq 0$. For instance, with J as in (6.3), we can take $A_0 = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}$ and $A_{\ell+1} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \end{pmatrix}^T$. These bookend vectors are the same for both branching programs, i.e., $A'_0 = A_0$ and $A'_{\ell+1} = A_{\ell+1}$.

[10] As J has order 2, the resulting product will still be the identity matrix.

These two branching programs $\mathbf{A}$ and $\mathbf{A}'$ are equivalent as they both compute the always zero function. Now, take $x = 0 \dots 0$ and $y = 0 \dots 010 \dots 0$ where the 1 is at the s-th position, and let $j = j_1$. Let us compute $(\tilde{a}_{x,j}, \tilde{a}_{y,j})$ for branching program $\mathbf{A}$ and $(\tilde{a}'_{x,j}, \tilde{a}'_{y,j})$ for branching program $\mathbf{A}'$.

Branching program A. As all matrices are identity matrices in $\mathbf{A}$, exchanging two matrices does not change the product and we have

$$\tilde{a}_{x,j} = A_0 \cdot \left(\prod_{i<j} A_{i,x[\mathtt{inp}(i)]} \right) \cdot A_{j,y[\mathtt{inp}(j)]} \cdot \left(\prod_{j<i\leq\ell} A_{i,x[\mathtt{inp}(i)]} \right) \cdot A_{\ell+1} = A_0 \cdot I \cdot A_{\ell+1} = 0,$$

$$\tilde{a}_{y,j} = A_0 \cdot \left(\prod_{i<j} A_{i,y[\mathtt{inp}(i)]} \right) \cdot A_{j,x[\mathtt{inp}(j)]} \cdot \left(\prod_{j<i\leq\ell} A_{i,y[\mathtt{inp}(i)]} \right) \cdot A_{\ell+1} = A_0 \cdot I \cdot A_{\ell+1} = 0.$$

Branching program $\mathbf{A}'$. Here, we chose our parameters so that a honest evaluation of $\mathbf{A}'$ on x leads to a product of only I matrices and a honest evaluation of $\mathbf{A}'$ on y leads to a product of $\ell-2$ matrices I and 2 matrices J. We also chose j so that we exchange a J matrix with an I matrix. Hence, we have

$$\tilde{a}'_{x,j} = A'_0 \cdot \left(\prod_{i<j} A'_{i,x[\mathtt{inp}(i)]} \right) \cdot A'_{j,y[\mathtt{inp}(j)]} \cdot \left(\prod_{j<i\leq\ell} A'_{i,x[\mathtt{inp}(i)]} \right) \cdot A'_{\ell+1} = A_0 \cdot J \cdot A_{\ell+1} \neq 0,$$

$$\tilde{a}'_{y,j} = A'_0 \cdot \left(\prod_{i<j} A'_{i,y[\mathtt{inp}(i)]} \right) \cdot A'_{j,x[\mathtt{inp}(j)]} \cdot \left(\prod_{j<i\leq\ell} A'_{i,y[\mathtt{inp}(i)]} \right) \cdot A'_{\ell+1} = A_0 \cdot J \cdot A_{\ell+1} \neq 0.$$

To conclude, this gives us the desired condition of Section 6.3.3. Indeed, for the branching program $\mathbf{A}$, the numerator of $\tilde{u}_{x,j} \cdot \tilde{u}_{y,j}$ is a multiple of g^2, hence zero-testing it with the parameter p'_{zt} gives a positive result. Oppositely, for the branching program $\mathbf{A}'$, the numerator of $\tilde{u}_{x,j} \cdot \tilde{u}_{y,j}$ is non-zero modulo g, hence zero-testing it with the parameter p'_{zt} gives a negative result. We can then distinguish between the obfuscations of $\mathbf{A}$ and $\mathbf{A}'$. This completes the proof of Theorem 6.5 for the obfuscators of [PST14, AGIS14, MSW14, GMM+16].

6.3.5 Other branching program obfuscators

We now discuss the possible extension of this attack to other branching program obfuscators that are not captured by the abstract obfuscator of Section 6.2.

6.3.5.1 Obfuscator of [BGK+14].

This obfuscator is close to the one described in the abstract model, except that it obfuscates a slightly different definition of branching programs. In [BGK+14], a branching program $\mathbf{A}$ comes with an additional ring element q_{acc}, and we have $\mathbf{A}(x) = 0$ if and only if $A_0 \cdot \prod_{i\in[l]} A_{i,x[\mathtt{inp}(i)]} \cdot A_{\ell+1} = q_{acc}$. The only difference with the definition of branching programs given in Section 2.6 is that q_{acc} may be non-zero. Hence, when multiplying by the scalars $\alpha_{i,\mathbf{b}}$ in the obfuscator (see Tool 3), we may change the output of the function. To enable correct evaluation of the obfuscated branching program, the obfuscator of [BGK+14] also publishes encodings of the scalars $\alpha_{i,\mathbf{b}}$ at level $\boldsymbol{v}_{i,\mathbf{b}}$.

More formally, the obfuscator of [BGK+14] uses Tools 2 and 3 of Section 6.2. In Tool 2, the authors choose matrices R_i which are invertible modulo g and use $R_{i+1}^{-1} \bmod g$ instead of R_{i+1}^{adj}, in order to keep the same product (otherwise the product would be multiplied by the determinants of the R_i matrices). Let $\hat{A}_{i,\mathbf{b}} = \alpha_{i,\mathbf{b}} R_i A_{i,\mathbf{b}} R_{i+1}^{-1}$ be the matrices obtained after re-randomization (using Tools 2 and 3). Let $\hat{A}_0 = A_0 R_1^{-1}$ and $\hat{A}_{\ell+1} = R_{\ell+1} A_{\ell+1}$. The obfuscator provides encodings of the matrices $\hat{A}_0$, $\{\hat{A}_{i,\mathbf{b}}\}_{i,\mathbf{b}}$ and $\hat{A}_{\ell+1}$ at levels $\boldsymbol{v}_0, \{\boldsymbol{v}_{i,\mathbf{b}}\}_{i,\mathbf{b}}$ and $\boldsymbol{v}_{\ell+1}$, respectively. It also provides encodings of the $\{\alpha_{i,\mathbf{b}}\}_{i,\mathbf{b}}$ at levels $\{\boldsymbol{v}_{i,\mathbf{b}}\}_{i,\mathbf{b}}$ and an encoding of q_{acc} at level $\boldsymbol{v}_0 + \boldsymbol{v}_{\ell+1}$. Then, to evaluate the obfuscated branching program on input x, one computes

$$[\hat{A}_0]_{\boldsymbol{v}_0} \cdot \prod_{i\in[\ell]} [\hat{A}_{i,x[\mathtt{inp}(i)]}]_{\boldsymbol{v}_{i,x[\mathtt{inp}(i)]}} \cdot [\hat{A}_{\ell+1}]_{\boldsymbol{v}_{\ell+1}} - [q_{acc}]_{\boldsymbol{v}_0+\boldsymbol{v}_{\ell+1}} \cdot \prod_{i\in[\ell]} [\alpha_{i,x[\mathtt{inp}(i)]}]_{\boldsymbol{v}_{i,x[\mathtt{inp}(i)]}},$$

and tests whether this is an encoding of 0 or not. By construction, this will be an encoding of 0 at level $\boldsymbol{v}^*$ if and only if $\mathbf{A}(x) = 0$.

The first part of our attack (recovering h and p'_{zt}) still goes through. We slightly modify the mixed-input part. Instead of exchanging only the j-th matrix between the evaluations of x and y, we also exchange the corresponding $\alpha_{j,\mathbf{b}}$ in the second product. Doing so, we ensure that the product of the $\alpha_{i,\mathbf{b}}$'s remains the same in both sides of the difference. This also ensures that the level of both sides is the same after the exchange, and hence we can still subtract them. The same example as in Section 6.3.4 then also works for this obfuscator. This gives us a way to distinguish in quantum polynomial time between the obfuscated versions of two equivalent branching programs, hence attacking the iO security of the obfuscator of [BGK+14].

6.3.5.2 Obfuscators of [GGH+13b, BR14].

Our attack does not seem to extend to the obfuscators of [GGH+13b, BR14]. The obstacle is that the security of these obfuscators against mixed-input attacks does not rely on the GGH13 map but on the scalars $\alpha_{i,\mathbf{b}}$, which are chosen with a specific structure to ensure that the branching program is correctly evaluated.

More precisely, these obfuscators use (single input) branching programs with a slightly different definition, where the product of matrices (with the bookend vectors) is never 0. For instance, the branching programs are chosen such that the product of the matrices (on honest evaluations) is either 1 or 2, in which cases we say that the output of the branching program is respectively 0 or 1. Hence, when evaluating the obfuscated branching program on input x, the user obtains a top-level encoding of either $\prod_i \alpha_{i,x_i}$ or $2\prod_i \alpha_{i,x_i}$ depending on the output of the branching program. In order for the user to determine which one of the two encodings it has obtained, the obfuscated branching program also provides him (via a so-called dummy branching program) with a top-level encoding $\prod_i \alpha_{i,x_i}$. The user then only has to subtract the two top-level encodings and zero-test to determine whether $\mathbf{A}(x) = 0$ or 1. Now, if the user tries to mix the inputs when evaluating the obfuscated branching program, it can obtain a top-level encoding of $(\alpha_{j,y_j} \cdot \prod_{i\neq j} \alpha_{i,x_i}) \cdot a_{x,j}$ for instance (where $a_{x,j} \in \{1, 2\}$ is the product of the corresponding matrices). However, as it is not a honest evaluation, it will not have a top-level encoding of $\alpha_{j,y_j} \cdot \prod_{i\neq j} \alpha_{i,x_i}$ to compare it with.

Following the same idea as for the mixed-input attack described above, the attacker could compute two top-level encodings of $(\alpha_{j,y_j} \cdot \prod_{i\neq j} \alpha_{i,x_i}) \cdot a_{x,j}$ and $(\alpha_{j,x_j} \cdot \prod_{i\neq j} \alpha_{i,y_i}) \cdot a_{y,j}$ and then multiply them to obtain an encoding of $(\prod_i \alpha_{i,x_i} \cdot \prod_i \alpha_{i,y_i}) \cdot a_{x,j} \cdot a_{y,j}$ at level $2\boldsymbol{v}^*$. Now, using the top-level encodings of $\prod_i \alpha_{i,x_i}$ and $\prod_i \alpha_{i,y_i}$ that are provided by the obfuscated branching program, one can also obtain an encoding of $(\prod_i \alpha_{i,x_i} \cdot \prod_i \alpha_{i,y_i})$ at level $2\boldsymbol{v}^*$. So if we could zero-test at level $2\boldsymbol{v}^*$, then we could distinguish between a branching program where $a_{x,j} \cdot a_{y,j} = 1$ and one where $a_{x,j} \cdot a_{y,j} \neq 1$. However, we cannot zero-test at level $2\boldsymbol{v}^*$: our new zero-testing parameter p'_{zt} only enables us to determine whether the numerator of an encoding is a multiple of g^2 or not. Here, we subtract two level-$2\boldsymbol{v}^*$ encodings of the same value, so the numerator of the result will be a multiple of g, but it is very unlikely to be a multiple of g^2. Hence, this is not clear whether we learn anything by using p'_{zt}. Because of the final subtraction, we did not manage to obtain an encoding at level $2\boldsymbol{v}^*$ whose numerator was a multiple of g^2, and so we did not manage to adapt the mixed-input attack described above to the obfuscators of [GGH+13b, BR14].

6.4 Conclusion

In this chapter, we have defined indistinguishability obfuscators and given a brief overview of the different techniques which have been used to construct candidate iOs. We have then described an abstract obfuscator, capturing the main techniques used in branching program obfuscation. Finally, we have presented a quantum attack against some candidate branching program obfuscators using the GGH13 map. This quantum attack only applies to the recent candidates, which use straddling set systems to prevent mixed-input attacks. Interestingly, these candidates have stronger security proofs in idealized models, but it makes them rely more on the underlying multilinear map and so makes them more vulnerable to weaknesses of this multilinear map.

A natural question raised by this quantum attack is about the quantum security of the GGHRSW

obfuscator. As mentioned above, the simple mixed-input attack described in this chapter does not seem to extend to the GGHRSW obfuscator, because of the final subtraction before the zero-test. However, being able to (double) zero-test at level $2\boldsymbol{v}^*$ gives the attacker a serious advantage, and a different attack using the double-zero-test could be possible against the GGHRSW obfuscator. Another question arises about the quantum aspect of this attack. We have seen that what is really required for the attack is the ability to efficiently find a somehow small element of a principal ideal. For the usual parameters of the GGH13 map, this can be done in polynomial quantum time, because the ideal has a short generator. If one wants to replace the quantum component by a classical one, then the run time would be $2^{O(\sqrt{n})}$, because one needs to first compute a generator of the principal ideal. By choice of the parameters of the GGH13 map, this gives a $2^{O(\lambda)}$ classical attack, which is not interesting. However, improvements for solving ideal-SVP classically with a somehow small approximation factor could translate into a classical attack against the obfuscators mentioned above, if the approximation factor is small enough (see Section 6.3.2).

Chapter 7

Conclusion

The contributions presented in this thesis can be sorted into two main categories: the ones related to fundamental problems over structured lattices (Chapters 3 and 4) and the ones related to the security of the GGH13 map and the obfuscators based on it (Chapters 5 and 6). In this conclusion, we come back to these two aspects of the thesis. We summarize what has been done and present some future directions that could be explored.

7.1 Ideal and module lattices

In Chapters 3 and 4, we have presented two algorithms finding short vectors in ideal and module lattices. These algorithms are theoretical results and show that the shortest vector problem might be easier to solve in structured lattices than in general ones. However, both algorithms suffer from a very strong limitation: the need for a CVP solver in a lattice L depending only on the number field. Recall that in the ideal lattice case, we can solve these CVP queries quite efficiently after an exponential pre-processing. In the module case however, we require much smaller solutions for the CVP problem and hence had to assume the existence of an oracle solving CVP in L. Recall also that in this case, the lattice has a dimension roughly n^2 (for prime power cyclotomic fields), which makes the situation even worse. These pre-processing/oracle requirements make our algorithm unusable in practice: if we want to run the pre-processing phase or implement the oracle, our algorithms become much slower than a BKZ call to solve approx-SVP in the original ideal/module lattice. Because the running time needed to run our algorithms in practice is so long, it also makes it very difficult to test the heuristics used in the algorithms. In the case of ideal lattices, we were able to test our heuristics up to number fields of degree roughly 35. In the module case, we were not even able to run a CVP solver on the lattice L for interesting number fields. Moreover, even if the experiments were consistent with the heuristics for the small dimensions we tested, it could be that this is only due to the smallness of the lattices involved.

One obvious interesting question related to these algorithms would be to remove these pre-processing and oracle requirements. To do so, one would need to analyze the lattices L involved in the algorithms and try to design a good CVP solver for these lattices. When trying to do so, it might be worth noticing that the lattices appearing in both algorithms can be randomly generated. Hence, instead of generating the lattice first and then trying to find an efficient CVP solver for it, one may try to sample the basis vectors of the lattice one by one to design a good lattice in which CVP can be solved easily. Finding an efficient CVP solver for the lattices involved in the algorithms, however, seems a difficult question. One of the reasons for that is that these lattices involve the log unit lattice and the lattice of relations of the class group, which seem difficult to analyze for an arbitrary number field. Moreover, the CVP solver we want to achieve should have a very small approximation factor (constant in the case of module lattices and smaller than $\sqrt{n}$ for ideal lattices). Hence, a solution to this question may be out of reach for the moment. Below, we describe simpler objectives, which should hopefully be easier to achieve, and which might help us solve the previous question.

Removing the heuristics. Removing the heuristics present in the two algorithms would be an interesting target. Recall that the algorithms rely on many heuristics, some of them are present in previous works and some of them are new. The ones present in previous works have already been studied and do not seem very problematic. The ones introduced for these algorithms however are more unsatisfactory. In particular, the fact that we cannot test them except for very small dimensions does not allow to

increase our confidence in them. Both chapters include a heuristic related to the covering radius of the lattice L introduced for the algorithm (or the distance of specific target points to the lattice in the case of modules). These heuristics are hard to check in practice (especially in the module case where the lattice has dimension roughly n^2) and are also difficult to justify mathematically. The justification we currently provide assumes that when one picks a randomly chosen ideal I of small algebraic norm and decomposes it in the class group as $I = \prod_j \mathfrak{p}_j^{\alpha_j} \cdot \langle g \rangle$, for some $\mathfrak{p}_j$ generating the class group, then the elements α_j and g are somehow randomly distributed. This assumption is quite plausible but remains heuristic. In an on-going work, de Boer, Ducas and Wesolowski [BDW19] are able to prove a similar statement, using Arakelov's divisors. This result could be a promising way to remove the more annoying heuristics of our algorithms (or at least give us more confidence in them).

Switching the defining polynomial. If we cannot get rid of the pre-processing phase or the oracle, we may then wonder whether it would be possible to reuse it for different number fields. If we managed to have a unique lattice for multiple number fields, then the pre-computations of the ideal-SVP solver would have to be done only once. Also, recall that finding a fast CVP solver in the general case seemed hard, but there may be some number fields for which the lattice L has a much nicer structure, which allows to efficiently solve CVP in it. One could for instance think about multiquadratics fields, which have a very strong algebraic structure, and which have already proven to be better than other number fields for solving some computational problems (recall that finding a generator of a principal ideal in multiquadratic fields can be done efficiently with a classical computer [BBV$^+$17], whereas we do not know how do to that for general number fields). With the current version of the algorithms, being able to find one good lattice L in which we can efficiently solve CVP only enables us to solve SVP in ideal/modules of the corresponding number field. Can we reuse the same lattice for different number fields?

One direction to try to link ideal (or module)-SVP between different number fields could be as follows. Recall that a number field is defined as $\mathbb{Q}[X]/P(X)$ for some irreducible polynomial P. Changing slightly some coefficients of P could drastically change the algebraic properties of the field. For instance, the NTRUPrime polynomial $P(X) = X^p - X - 1$ (where p is prime) only differs by one coefficient from the multiple $X^p - 1$ of the cyclotomic polynomial $(X^p - 1)/(X - 1)$, but its algebraic properties (automorphisms, sub-fields...) differ a lot [BCLV17]. On the other hand, changing just a few coefficients of the defining polynomial might move only slightly the roots of the polynomial, and the geometric properties of the number field only depends on these roots. Hence, by changing carefully some coefficients of the defining polynomial, one may end up with a new number field with completely different algebraic properties but similar geometric properties. Hence, when working in a number field K_1 whose algebraic properties are not nice, one may try to find a new number field K_2 with better algebraic properties and similar geometry. Assume that for the number field K_2, the algebraic properties of the field enable us to have a fast CVP solver for our lattice L. Then we could efficiently solve ideal-SVP in K_2 and then come back to K_1. Because "being small" is a geometric property, one may hope that we still have a short vector when we come back to K_1. Note that this idea of switching polynomials could have more applications than simply reusing the lattice L between different number fields.

7.2 The GGH13 map and obfuscators

We have presented in this thesis two attacks against the GGH13 map and its applications. One of the attacks uses the statistical properties of the GGH13 map (see Chapter 5), while the other relies on the algebraic properties of the GGH13 map, and the ideal lattices used in it (see Chapter 6). Looking at the current status of candidate obfuscators, one may observe that there still remain unbroken obfuscators based on the GGH13 map (see Section 6.1.2). However, few constructions based on the GGH13 map have been proposed recently (the last one was the GMMSSZ obfuscator in 2016 [GMM$^+$16]).[1] Even if the GGH13 map is not completely broken, the high number of attacks against obfuscators using it seems to have decreased the confidence of the community in its security. The recent constructions of obfuscators have mostly used the GGH15 map, or have gone via functional encryption, for which fewer attacks are known. It seems that in the upcoming years, the interest of cryptographers for the GGH13

[1] This may not seem a very long time ago, but it should be kept in mind that the first candidate obfuscator was proposed in 2013 [GGH$^+$13b], and that between 2013 and 2016, more than 7 candidates based on the GGH13 map were proposed.

map and CLT13 map may be replaced by the GGH15 map and obfuscation via functional encryption. From a cryptanalytic point of view, the GGH13 map has been widely studied and we know many weaknesses which can potentially be used for attacks. Hence, even if the GGH13 map falls out of favor for constructing obfuscators, the attacks we know against it can give us some insight on the security of the other constructions.

Extending the known attacks against the GGH13 map. The two candidate multilinear maps CLT13 and GGH13 share a similar structure. However, the main categories of attacks known against these maps use different techniques. The GGH13 map is subject to annihilation attacks, which recover a secret principal ideal of the map [MSZ16], whereas the CLT13 map is subject to input-partitionable attacks, whose aim is to recover some secret elements by computing the eigenvalues of a matrix [CHL$^+$15, CLLT17]. Techniques related to input-partitionable attacks have been used by Chen et al. [CGH17], combined with annihilation techniques, to attack the GGHRSW obfuscator based on the GGH13 map. Hence, it seems that the techniques used to attack CLT13-based constructions can also be used against the GGH13 map. However, the converse is not known. It would be interesting to try to adapt the annihilation attacks of the GGH13 map to the CLT13 one, especially because these attacks are more devastating than the one usually applied in the CLT13 context. Indeed, the input-partitionable attacks can be prevented by the FRS technique [FRS17], whereas we do not have any patch for the annihilation attacks. One may also want to try to adapt the study of the statistical attacks presented in this thesis to the CLT13 setting. In the case of the CLT13 map, we are working with integers instead of polynomials, hence it might be easier to exploit the leakage and mount an attack against the map.

Because of the current interest in the GGH15 map, one might also be interested in trying to adapt the GGH13 attacks against the GGH15 map. The GGH13 and GGH15 maps are less similar than the GGH13 and CLT13 maps. However, the recent work of Cheon et al. [CCH$^+$18] shows that statistical attacks are possible against the GGH15 map. Maybe the annihilation attacks could be adapted too.

New assumptions. The line of work trying to construct obfuscation from functional encryption has received a lot of attention in the past year, and it is now known that obfuscation can be constructed from some variants of the RLWE problem, where information about the noise of the RLWE instances is leaked. These new variants of RLWE have been introduced very recently and so assessing their security is still a challenge. Can we show that these variants are harder to solve than the plain RLWE problem? If we cannot, can we show that they are harder to solve than module-SIVP or ideal-SVP? Or, on the contrary, can we attack them? These RLWE variants imply obfuscation, hence, proving or disproving their difficulty would be an interesting question.

Post-quantum obfuscation. The current status of candidate obfuscators is quite complicated, with constructions, attacks, partial attacks and patches. However, when one is interested in post-quantum obfuscation, the number of remaining candidates is drastically reduced. It would be interesting to try to assess the quantum security of the remaining candidate obfuscators. Because many candidates use lattices techniques, the presence of a quantum attack against these candidates is a bad sign for the security of the obfuscator. Indeed, it implies in particular that the construction cannot be proved secure under standard lattice assumptions (which are believed to be post-quantum). Hence, the search for post-quantum obfuscators would provide both obfuscators resistant to quantum computers, but also obfuscators which are more likely to be secure under standard assumptions.

List of publications

[DP18] Léo Ducas and Alice Pellet-Mary. On the statistical leak of the GGH13 multilinear map and some variants. In *Advances in Cryptology – ASIACRYPT*, pages 465–493. Springer, 2018. 16, 24, 82

[Pel18] Alice Pellet-Mary. Quantum attacks against indistinguishablility obfuscators proved secure in the weak multilinear map model. In *Advances in Cryptology – CRYPTO*, pages 153–183. Springer, 2018. 14, 16, 21, 24, 105, 109, 111, 113

[PHS19] Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. Approx-SVP in ideal lattices with pre-processing. In *Advances in Cryptology – EUROCRYPT*, pages 685–716. Springer, 2019. 14, 22, 41, 61, 62

[LPSW19] Changmin Lee, Alice Pellet-Mary, Damien Stehlé, and Alexandre Wallet. An LLL algorithm for module lattices. Accepted at Asiacrypt 2019. 16, 23, 57

Bibliography

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *Theory of Cryptography Conference – TCC*, pages 528–556. Springer, 2015. 113, 116, 151, 153, 155

[ABD16] Martin Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched ntru assumptions. In *Advances in Cryptology – CRYPTO*, pages 153–178. Springer, 2016. 89, 115

[AD17] Martin R. Albrecht and Amit Deo. Large modulus ring-LWE $\geq$ module-LWE. In *Advances in Cryptology – ASIACRYPT*, volume 10624 of *Lecture Notes in Computer Science*, pages 267–296. Springer, 2017. 13, 21, 45, 57, 58

[ADGM17] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over ggh13. In *44th International Colloquium on Automata, Languages, and Programming – ICALP*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. 90, 109, 110

[AGIS14] Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 646–658. ACM, 2014. 109, 110, 111, 112, 114, 115, 116, 117, 119, 120, 125

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In *Advances in Cryptology – EUROCRYPT*. Springer, 2019. 114

[AJS18] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: io from lwe, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. `http://eprint.iacr.org/2018/615`. 114

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, pages 99–108. ACM, 1996. 11, 19, 58

[Ajt98] Miklós Ajtai. The shortest vector problem in l_2 is NP-hard for randomized reductions. In *STOC*, 1998. 61

[AKS01] Miklós Ajtai, Ravi Kumar, and Dandapani Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610. ACM, 2001. 10, 18

[AS18] Divesh Aggarwal and Noah Stephens-Davidowitz. Just take the average! an embarrassingly simple 2^n-time algorithm for svp (and cvp). In *Symposium on Simplicity in Algorithms – SOSA*, 2018. 10, 18

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. 27

[Bac90] Eric Bach. Explicit bounds for primality testing and related problems. *Mathematics of Computation*, 55(191):355–380, 1990. 32

[BBV+17] Jens Bauch, Daniel J. Bernstein, Henry de Valence, Tanja Lange, and Christine van Vredendaal. Short generators without quantum computers: the case of multiquadratics. In *Advances in Cryptology – EUROCRYPT*, pages 27–59. Springer, 2017. 35, 56, 130

[BCLV17] Daniel J Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. Ntru prime: reducing attack surface at low cost. In *International Conference on Selected Areas in Cryptography – SAC*, pages 235–260. Springer, 2017. 130

[BD16] Zvika Brakerski and Or Dagmi. Shorter circuit obfuscation in challenging security models. In *International Conference on Security and Cryptography for Networks*, pages 551–570. Springer, 2016. 113, 116, 151, 153, 155

[BDW19] Koen de Boer, Léo Ducas, and Benjamin Wesolowski, 2019. Personal communication. 130

[BEF+17] Jean-François Biasse, Thomas Espitau, Pierre-Alain Fouque, Alexandre Gélin, and Paul Kirchner. Computing generator in cyclotomic integer rings. In *Advances in Cryptology – EUROCRYPT*, pages 60–88. Springer, 2017. 35, 42, 43, 45, 46, 55, 59, 89, 115

[Ber14] Daniel J. Bernstein. A subfield-logarithm attack against ideal lattices: Computational algebraic number theory tackles lattice-based cryptography. The cr.yp.to blog, 2014. `https://blog.cr.yp.to/20140213-ideal.html`. 42

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology – CRYPTO*, pages 213–229. Springer, 2001. 82

[BF14] Jean-François Biasse and Claus Fieker. Subexponential class group and unit group computation in large degree number fields. *LMS Journal of Computation and Mathematics*, 17(A):385–403, 2014. 34, 35, 43, 45, 46, 55, 89

[BFH17] Jean-François Biasse, Claus Fieker, and Tommy Hofmann. On the computation of the hnf of a module over the ring of integers of a number field. *Journal of Symbolic Computation*, 80:581–615, 2017. 31, 61

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Advances in Cryptology – CRYPTO*, pages 1–18. Springer, 2001. 85, 106, 107, 114

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology – EUROCRYPT*, pages 221–238. Springer, 2014. 85, 109, 110, 111, 112, 114, 115, 116, 119, 120, 125, 126

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of ggh15: Provable security against zeroizing attacks. In *Theory of Cryptography Conference – TCC*, pages 544–574. Springer, 2018. 112, 113

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory – TOCT*, 6(3):13, 2014. 12, 20, 58

[Bia17] Jean-François Biasse. Approximate short vectors in ideal lattices of $\mathbb{Q}(\zeta_{p^e})$ with precomputation of $\mathrm{Cl}(\mathcal{O}_k)$. In *International Conference on Selected Areas in Cryptography – SAC*, pages 374–393. Springer, 2017. 35, 56

[BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In *Advances in Cryptology – EUROCRYPT*, pages 764–791. Springer, 2016. 112

[BP91] Wieb Bosma and Michael Pohst. Computations with finitely generated modules over dedekind rings. In *International symposium on Symbolic and algebraic computation – ISSAC*, pages 151–156. ACM, 1991. 31, 61

[BR13] Zvika Brakerski and Guy N. Rothblum. Obfuscating conjunctions. pages 416–434, 2013. 85

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography Conference – TCC*, pages 1–25. Springer, 2014. 85, 108, 109, 110, 111, 112, 114, 116, 126

[BS96] Eric Bach and Jeffrey Outlaw Shallit. *Algorithmic Number Theory: Efficient Algorithms*, volume 1. MIT press, 1996. 32

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003. 82, 85

[BS16] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM symposium on Discrete algorithms – SODA*, pages 893–902. Society for Industrial and Applied Mathematics, 2016. 34, 35, 42, 43, 59, 115, 120

[Buc88] Johannes Buchmann. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. *Séminaire de théorie des nombres, Paris*, 1989(1990):27–41, 1988. 44, 46

[Buc94] Johannes Buchmann. Reducing Lattice Bases by Means of Approximations. In *International Algorithmic Number Theory Symposium – ANTS*, 1994. 36

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE Computer Society, 2011. 11, 19

[BV18] Jean-François Biasse and Christine van Vredendaal. Fast multiquatratic S-unit computation and application to the calculation of class groups. In *International Algorithmic Number Theory Symposium – ANTS*. Springer, 2018. 35, 56

[BVWW16] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring lwe. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 147–156. ACM, 2016. 114

[BZ17] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *Algorithmica*, 79(4):1233–1285, 2017. 107

[CCH+18] Jung Hee Cheon, Wonhee Cho, Minki Hhan, Jiseung Kim, and Changmin Lee. Statistical zeroizing attack: Cryptanalysis of candidates of bp obfuscation over ggh15 multilinear map. Cryptology ePrint Archive, Report 2018/1081, 2018. `http://eprint.iacr.org/2018/1081`. 112, 113, 131

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In *Advances in Cryptology – EUROCRYPT*, pages 559–585. Springer, 2016. 16, 24, 33, 35, 42, 44, 45, 46, 55, 115, 120

[CDW17] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-svp. In *Advances in Cryptology – EUROCRYPT*, pages 324–348. Springer, 2017. 14, 22, 42, 44, 45, 46, 55, 61, 62

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New mmap attacks and their limitations. In *Advances in Cryptology – CRYPTO*, pages 247–266. Springer, 2015. 93, 112, 113

[CGH17] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In *Advances in Cryptology – EUROCRYPT*, pages 278–307. Springer, 2017. 14, 21, 90, 92, 109, 110, 111, 112, 131

[CGS14a] Peter Campbell, Michael Groves, and Dan Shepherd. Soliloquy: A cautionary tale. In *ETSI 2nd Quantum-Safe Crypto Workshop*, pages 1–9, 2014. 35, 115, 120

[CGS14b] Peter Campbell, Michael Groves, and Dan Shepherd. Soliloquy: A cautionary tale, 2014. Available at `http://docbox.etsi.org/Workshop/2014/201410_CRYPTO/S07_Systems_and_Attacks/S07_Groves_Annex.pdf`. 42

[CHKL18] Jung Hee Cheon, Minki Hhan, Jiseung Kim, and Changmin Lee. Cryptanalyses of branching program obfuscations over ggh13 multilinear map from the ntru problem. In *Advances in Cryptology – CRYPTO*, pages 184–210. Springer, 2018. 14, 21, 109, 110, 111

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology – EUROCRYPT*, pages 3–12. Springer, 2015. 86, 112, 131

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016. 89, 115

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of ggh15 multilinear maps. In *Advances in Cryptology – CRYPTO*, pages 607–628. Springer, 2016. 86

[CLLT17] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over clt13. In *IACR International Workshop on Public Key Cryptography – PKC*, pages 41–58. Springer, 2017. 110, 111, 112, 131

[CLT13] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology – CRYPTO*, pages 476–493. Springer, 2013. 86

[Coh95] Henri Cohen. *A Course in Computational Algebraic Number Theory*. 1995. 36

[Coh96] Henri Cohen. Hermite and Smith normal form algorithms over Dedekind domains. 1996. 31, 61

[Coh13] Henri Cohen. *A course in computational algebraic number theory*, volume 138. Springer Science & Business Media, 2013. 32, 33

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. Ggh15 beyond permutation branching programs: Proofs, attacks, and candidates. In *Advances in Cryptology – CRYPTO*, pages 577–607. Springer, 2018. 112, 113

[DGG+18] Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee. Obfuscation from low noise multilinear maps. In *International Conference in Cryptology in India – Indocrypt*, pages 329–352. Springer, 2018. 91, 92, 93, 94, 95, 96, 102, 113, 116, 145, 147, 151, 153, 155

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976. 9, 17

[DLW19] Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. Finding closest lattice vectors using approximate Voronoi cells. In *PQCRYPTO*. Springer, 2019. To appear. 45

[DPW19] Léo Ducas, Maxime Plançon, and Benjamin Wesolowski. On the shortness of vectors to be found by the ideal-svp quantum algorithm. In *Advances in Cryptology – CRYPTO*. Springer, 2019. 42

[EHKS14] Kirsten Eisenträger, Sean Hallgren, Alexei Kitaev, and Fang Song. A quantum algorithm for computing the unit group of an arbitrary degree number field. In *STOC*, pages 293–302. ACM, 2014. 42

[FHHL18] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. Graded encoding schemes from obfuscation. In *IACR International Workshop on Public Key Cryptography – PKC*, pages 371–400. Springer, 2018. 107

[Fie97] Claus Fieker. *Über relative Normgleichungen in älgebraischen Zahlkörpern.* PhD thesis, TU Berlin, 1997. 58

[FP96] Claus Fieker and ME Pohst. On lattices over number fields. In *International Algorithmic Number Theory Symposium – ANTS*, pages 133–139. Springer, 1996. 58, 61

[FP06] Claus Fieker and Michael Pohst. Dependency of units in number fields. *Mathematics of Computation*, 75(255):1507–1518, 2006. 33

[FRS17] Rex Fernando, Peter MR Rasmussen, and Amit Sahai. Preventing clt attacks on obfuscation with linear overhead. In *Advances in Cryptology – ASIACRYPT*, pages 242–271. Springer, 2017. 109, 110, 111, 112, 131

[FS10] Claus Fieker and Damien Stehlé. Short bases of lattices over number fields. In *International Algorithmic Number Theory Symposium – ANTS*, pages 157–173. Springer, 2010. 31, 36, 59, 78

[Gel17] Alexandre Gelin. *Calcul de groupes de classes d'un corps de nombres et applications à la cryptologie.* PhD thesis, Paris 6, 2017. 35, 46, 55

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009. 11, 19

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology – EUROCRYPT*, pages 1–17. Springer, 2013. 13, 16, 21, 24, 45, 82, 83, 84, 86, 87, 88, 89, 90, 91, 93, 94, 102, 114, 119, 120, 151

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. pages 40–49, 2013. 13, 21, 107, 108, 109, 110, 111, 116, 126, 130

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography Conference – TCC*, pages 498–527. Springer, 2015. 86, 112

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476. ACM, 2013. 107

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science – FOCS*, pages 612–621. IEEE, 2017. 86, 114

[GLM09] Ying Hung Gan, Cong Ling, and Wai Ho Mow. Complex lattice reduction algorithm for low-complexity full-diversity mimo detection. *Transactions on Signal Processing*, 57(7):2701–2710, 2009. 58

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In *Theory of Cryptography Conference – TCC*, pages 241–268. Springer, 2016. 13, 21, 92, 109, 110, 111, 114, 115, 116, 117, 119, 120, 125, 130, 145, 147

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008. 39

[GR07] Shafi Goldwasser and Guy N Rothblum. On best-possible obfuscation. In *Theory of Cryptography Conference – TCC*, pages 194–213. Springer, 2007. 107

[HB18] Máté Horváth and Levente Buttyán. The birth of cryptographic obfuscation–a survey. Technical report, 2018. `http://eprint.iacr.org/2015/412`. 108

[HJ16] Yupu Hu and Huiwen Jia. Cryptanalysis of ggh map. In *Advances in Cryptology – EUROCRYPT*, pages 537–565. Springer, 2016. 13, 16, 21, 24, 86, 90, 91, 93

[HM89] James L. Hafner and Kevin S. McCurley. A rigorous subexponential algorithm for computation of class groups. *Journal of the American mathematical society*, 2(4):837–850, 1989. 44, 46

[Hop98] Andreas Hoppe. *Normal forms over Dedekind domains, efficient implementation in the computer algebra system KANT*. PhD thesis, TU Berlin, 1998. 30

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium – ANTS*, pages 267–288. Springer, 1998. 13, 21, 58

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In *Advances in Cryptology – CRYPTO*, pages 447–464. Springer, 2011. 10, 18

[HWB17] Patrick Holzer, Thomas Wunderer, and Johannes A. Buchmann. Recovering short generators of principal fractional ideals in cyclotomic fields of conductor $p^\alpha q^\beta$. In *International Conference in Cryptology in India – Indocrypt*, pages 346–368. Springer, 2017. 42

[Jou00] Antoine Joux. A one round protocol for tripartite diffie–hellman. In *International Algorithmic Number Theory Symposium – ANTS*, pages 385–393. Springer, 2000. 82

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987. 60

[KF17] Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched ntru parameters. In *Advances in Cryptology – EUROCRYPT*, pages 3–26. Springer, 2017. 89, 115

[KL17] Taechan Kim and Changmin Lee. Lattice reductions over euclidean rings with applications to cryptanalysis. In *IMA International Conference on Cryptography and Coding*, pages 371–391. Springer, 2017. 59, 61

[Laa16] Thijs Laarhoven. Sieving for closest lattice vectors (with preprocessing). In *International Conference on Selected Areas in Cryptography – SAC*, pages 523–542. Springer, 2016. 26, 45, 54, 62

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Advances in Cryptology – EUROCRYPT*, pages 28–57. Springer, 2016. 114

[LLL82] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. 15, 23, 36, 37, 58

[LM00] Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000. 53

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *International Colloquium on Automata, Languages, and Programming – ICALP*, pages 144–155. Springer, 2006. 20, 58

[LM18] Huijia Lin and Christian Matt. Pseudo flawed-smudging generators and their application to indistinguishability obfuscation. Cryptology ePrint Archive, Report 2018/646, 2018. `http://eprint.iacr.org/2018/646`. 114

[Lou00] Stéphane Louboutin. Explicit bounds for residues of dedekind zeta functions, values of l-functions at $s = 1$, and relative class numbers. *Journal of Number Theory*, 85(2):263–282, 2000. 33

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology – EUROCRYPT*, pages 1–23. Springer, 2010. 12, 20, 42, 58

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015. 12, 20, 45, 57, 58

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. Gghlite: More efficient multilinear maps from ideal lattices. In *Advances in Cryptology – EUROCRYPT*, pages 239–256. Springer, 2014. 89

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective*. 2002. 61

[Mic01] Daniele Micciancio. The hardness of the closest vector problem with preprocessing. *Transactions on Information Theory*, 47(3):1212–1215, 2001. 60

[Min67] Hermann Minkowski. *Gesammelte Abhandlungen*. Chelsea, New York, 1967. 43

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007. 11, 19, 38, 39

[MSW14] Eric Miles, Amit Sahai, and Mor Weiss. Protecting obfuscation against arithmetic attacks. *IACR Cryptology ePrint Archive*, 2014:878, 2014. 109, 110, 111, 112, 114, 115, 116, 117, 119, 120, 125

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. In *Advances in Cryptology – CRYPTO*, pages 629–658. Springer, 2016. 13, 14, 16, 21, 24, 90, 92, 109, 110, 111, 114, 116, 131, 145, 146, 148

[MZ18] Fermi Ma and Mark Zhandry. The mmap strikes back: Obfuscation and new multilinear maps immune to clt13 zeroizing attacks. In *Theory of Cryptography Conference – TCC*, pages 513–543. Springer, 2018. 86, 111, 112

[Nap96] Huguette Napias. A generalization of the lll-algorithm over euclidean rings or orders. *Journal de théorie des nombres de Bordeaux*, 8(2):387–396, 1996. 58

[O'M63] Onorato Timothy O'Meara. *Introduction to Quadratic Forms*. Springer, 1963. 58

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography Conference – TCC*, pages 145–166. Springer, 2006. 20, 58

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In *STOC*, pages 461–473. ACM, 2017. 42

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology – CRYPTO*, pages 500–517. Springer, 2014. 109, 110, 111, 112, 114, 115, 116, 117, 119, 120, 125

[RBV04] Ghaya Rekaya, Jean-Claude Belfiore, and Emanuele Viterbo. A very efficient lattice reduction tool on fast fading channels. In *ISITA*, 2004. 42

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005. 11, 19, 42, 58

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 9, 17

[RSW18] Miruna Rosca, Damien Stehlé, and Alexandre Wallet. On the ring-LWE and polynomial-LWE problems. In *Advances in Cryptology – EUROCRYPT*, pages 146–173. Springer, 2018. 58

[Sam13] Pierre Samuel. *Algebraic Theory of Numbers: Translated from the French by Allan J. Silberger*. Courier Corporation, 2013. 33

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, 53:201–224, 1987. 10, 18, 42

[SD19] Noah Stephens-Davidowitz. A time-distance trade-off for gdd with preprocessing—instantiating the dlw heuristic. arXiv preprint arXiv:1902.08340, 2019. 45

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66:181–199, 1994. 10, 18, 41, 42, 58

[Sho97] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. 11, 19

[SMSV14] Saruchi, Ivan Morel, Damien Stehlé, and Gilles Villard. LLL reducing with the most significant bits. In *International Symposium on Symbolic and Algebraic Computation – ISSAC*, pages 367–374. ACM, 2014. 36, 60

[SS13] Damien Stehle and Ron Steinfeld. Making ntruenrypt and ntrusign as secure as standard worst-case problems over ideal lattices, 2013. `http://eprint.iacr.org/2013/004`. 95

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *Advances in Cryptology – ASIACRYPT*, pages 617–635. Springer, 2009. 12, 20, 42, 58

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484. ACM, 2014. 107

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science – FOCS*, pages 600–611. IEEE, 2017. 86, 114

[XZKL17] Hui Xu, Yangfan Zhou, Yu Kang, and Michael R Lyu. On secure and usable program obfuscation: A survey. arXiv preprint arXiv:1710.01139, 2017. 105

[Zim80] Rainer Zimmert. Ideale kleiner Norm in Idealklassen und eine Regulatorabschätzung. *Inventiones mathematicae*, 62(3):367–380, 1980. 33

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *Advances in Cryptology – EUROCRYPT*, pages 439–467. Springer, 2015. 113, 116, 151, 153, 155

List of Figures

List of Tables

List of Algorithms

Appendix A

Security proof of our simple setting in the weak multilinear map model

In this section, we first recall the weak multilinear map model for the GGH13 map (mentioned first in [MSZ16] and then used in [GMM+16] and in [DGG+18]). Then we prove that the setting we described in Section 5.3.2 is secure in the weak multilinear map model, where security is defined as in Definition 5.4.

A.1 The weak multilinear map model

The idea of the weak multilinear map model [MSZ16, GMM+16, DGG+18] is to limit the power of the attacker by not giving it the encoded values directly. Instead, an oracle $\mathcal{M}$ keeps a table with the encoded values and allows the attacker to perform only some operations on these encoded values. More formally, an encoded element is a couple $(a, \boldsymbol{v})$, with $a \in R/gR$ and $\boldsymbol{v} \in \{0,1\}^\kappa$. Recall that we denote by $\boldsymbol{v}^*$ the vector $(1, 1, \cdots, 1)$. We can perform the following arithmetic operations on the encoded elements:

- **Addition/subtraction.** For any $\boldsymbol{v}$, we have $(a, \boldsymbol{v}) \pm (b, \boldsymbol{v}) = (a \pm b, \boldsymbol{v})$.
- **Multiplication.** If $\boldsymbol{v}$ and $\boldsymbol{w}$ are such that $\boldsymbol{v}[i]\boldsymbol{w}[i] = 0$ for all $i \in \{1, \cdots, \kappa\}$, then we have $(a, \boldsymbol{v}) \cdot (b, \boldsymbol{w}) = (a \cdot b, \boldsymbol{v} + \boldsymbol{w})$.
- **Scalar multiplication.** For any $\boldsymbol{v} \in \{0,1\}^\kappa$, $a \in R/gR$ and $\alpha \in R$, we have $\alpha \cdot (a, \boldsymbol{v}) = (\alpha \cdot a, \boldsymbol{v})$.

The oracle $\mathcal{M}$ implements the following interfaces.

Initialization. The oracle first initializes the parameters. It sets n to be a power of 2, defines $R = \mathbb{Z}[X]/(X^n + 1)$ and samples g an element of R. The size of the parameters is the same as the one we described in Section 5.2.2. The oracle $\mathcal{M}$ then receives a set of r pairs $(a, \boldsymbol{v})$ to encode. It creates a table T in which it stores the pairs $(a, \boldsymbol{v})$ together with a handle h_i it generates, which is independent of the encoded value a but reveals the level of the encoding $\boldsymbol{v}$. Finally, the oracle outputs the handles h_i, for $1 \leq i \leq r$. The oracle $\mathcal{M}$ also creates a table T' for post-zero-test values, that is empty for the moment. This interface has to be called before the other ones, and any attempt to call this procedure more than once will fail.

Operations on encodings. Given two handles h_1, h_2 and an operation $\circ \in \{+, -, \times\}$, the oracle first checks whether the handles h_1 and h_2 are in its table. If one of them is not in the table, then in returns $\perp$. Otherwise, let $(a_1, \boldsymbol{v}_1)$ and $(a_2, \boldsymbol{v}_2)$ be the encoded elements associated to these handles. If $\circ \in \{+, -\}$, $\mathcal{M}$ checks whether $\boldsymbol{v}_1 = \boldsymbol{v}_2$. If this is not the case, $\mathcal{M}$ outputs $\perp$. Otherwise, it creates a new entry in its table, with the encoded value $(a_1 \circ a_2, \boldsymbol{v}_1)$ and a new handle h and it outputs h. If $\circ = \times$, then $\mathcal{M}$ checks whether $\boldsymbol{v}_1[i]\boldsymbol{v}_2[i] = 0$ for all $i \in \{1, \cdots, \kappa\}$. If this is not the case, it outputs $\perp$, otherwise it creates a new entry in its table with the encoded value $(a_1 \cdot a_2, \boldsymbol{v}_1 + \boldsymbol{v}_2)$ and a new handle h and it outputs h.

Multiplication by an element of R. Given a handle h and an element $\alpha \in R$, the oracle $\mathcal{M}$ first checks whether h is in its table T. If it is not, $\mathcal{M}$ outputs $\perp$. Otherwise, let $(a, \boldsymbol{v})$ be the corresponding encoded value. The oracle creates a new entry in its table, with encoded value $(\alpha a, \boldsymbol{v})$ and a new handle h'. Then it outputs h'.

Zero-test query. Given a handle h, the oracle $\mathcal{M}$ checks whether h is in its table. If not $\mathcal{M}$ outputs $\perp$. Otherwise let $(a, \boldsymbol{v})$ be the associated encoded value. If $\boldsymbol{v} \neq \boldsymbol{v}^*$, then $\mathcal{M}$ outputs $\perp$. Otherwise, the oracle checks whether a is a multiple of g. If it is, then $\mathcal{M}$ creates a new entry in its post-zero-test table T', with value $a/g \in R$ and with a new handle h', then it outputs h'. If a is not a multiple of g, then $\mathcal{M}$ outputs $\perp$.

Post-zero-test query. Given a polynomial p of degree polynomial in n,[1] and a bunch of handles $h_1, \cdots, h_t$, the oracle checks whether all handles h_i are in its post-zero-test table T'. If it is not the case, or if p is the identically zero polynomial, it outputs $\perp$. Otherwise, let r_i be the value corresponding to the handle h_i in T'. The oracle computes $p(r_1, \cdots, r_t)$. If this is a non zero multiple of g, then the adversary outputs "WIN", otherwise it outputs $\perp$.

The adversary wins the game if it manages to have the oracle output "WIN" after a post-zero-test query. We recall here the security definition given in Definition 5.4, adapted to the weak multilinear map model.

Definition A.1 (Security of our setting)**.** We say that our setting of the GGH multilinear map is secure in the weak multilinear map model if any polynomial time adversary has negligible probability to make the oracle output "WIN", when the oracle is initialized with the elements $a_{\boldsymbol{v},i}$ defined in Section 5.3.2.

We will now prove that our simple setting defined in Section 5.3.2 is secure in the weak multilinear map model, for the definition of security given above. This proof does not depend on the sampling method chosen, as long as it has enough min-entropy. As all the sampling methods described in Chapter 5 have enough min-entropy, our setting will be secure in the weak multilinear map model, independently of the sampling method chosen.

A.2 Mathematical tools

Definition A.2. Let Y be a random variable with values in a set S, the *guessing probability* of the variable Y is

$$\max_{s \in S} \Pr(Y = s)$$

and the *min-entropy* of Y is defined by

$$H_\infty(Y) = -\log(\max_{s \in S} \Pr(Y = s)).$$

For multiple random variables $Y_1, \cdots, Y_k$, we let $p_i(s_1, \cdots, s_{i-1})$ be the guessing probability of Y_i conditioned on $Y_j = s_j$ for $j < i$. Then, define $p_i = \mathbb{E}_{Y_1, \cdots, Y_{i-1}}[p_i(Y_1, \cdots, Y_{i-1})]$ to be the expectation of the $p_i(Y_1, \cdots, Y_{i-1})$. Finally, we denote by $p_{\max}(Y_1, \cdots, Y_k) = \max_i p_i$ the maximum of the p_i's.

For the proof of our theorem, we will use the improved Schwartz-Zippel lemma of [MSZ16, Section 5.1]. The classical Schwartz-Zippel lemma needs independent and uniform random variables, while this version allows us to use random variables that might be correlated and non uniform, as long as they have enough min-entropy. The statement of the lemma is the following:

Lemma A.3 (Improved Schwartz-Zippel lemma from [MSZ16])**.** *Let $\mathbb{F}$ be a field, $k > 0$ be an integer and $P \in \mathbb{F}[X_1, \cdots, X_k]$ be a polynomial of degree at most d. Let $Y_1, \cdots, Y_k$ be random variables in $\mathbb{F}$ (that might be correlated and non uniform). Then we have*

$$\Pr_{Y_1, \cdots, Y_k}[P(Y_1, \cdots, Y_k) = 0] \leq d \cdot p_{\max}(Y_1, \cdots, Y_k).$$

[1] This restriction on the degree of p may seem a bit unnatural, but this is needed for the proof, and it was already used (and discussed) in [MSZ16].

A.3 Security proof

The main idea of the security proof is to use the Schwartz-Zippel lemma to show that, except with negligible probability, the adversary will never be able to create encodings that pass the zero-test, except for linear combinations of the ones that are made public in the setting. Then, it is easy to show that the adversary cannot use these encodings to create a multiple of g. This kind of proof was already used to prove security of obfuscators in the weak multilinear map model [GMM+16, DGG+18]. As our setting is simpler than a candidate obfuscator, the proof will also be easier.

First, we recall and precise how we generate the encodings in our simple setting of the GGH map. Let D_a be a distribution over the elements of R that are invertible modulo g, with min-entropy at least n when reduced modulo g (i.e., for all $x \in (R/gR)^\times$, we have $\Pr_{y \leftarrow D_a}(y = x \mod g) \leq 2^{-n}$). Let also D_r be a distribution over R with min-entropy at least n.[2]

For all $\boldsymbol{v} \in \mathcal{A}$ of weight 1, sample m elements $a_{\boldsymbol{v},i} \leftarrow D_a$ independently (with $1 \leq i \leq m$) and $m-1$ elements $a_{\tilde{\boldsymbol{v}},1}, \cdots, a_{\tilde{\boldsymbol{v}},m-1} \leftarrow D_a$ independently. Then, sample $r_{\boldsymbol{v}} \leftarrow D_r$ and let

$$a_{\tilde{\boldsymbol{v}},m} = -\widehat{a}_{\boldsymbol{v},m} \sum_{i=1}^{m-1} a_{\boldsymbol{v},i} a_{\tilde{\boldsymbol{v}},i} + r_{\boldsymbol{v}} g$$

where $\widehat{a}_{\boldsymbol{v},m}$ is an element in R such that $\widehat{a}_{\boldsymbol{v},m} a_{\boldsymbol{v},m} = 1 \mod g$, i.e. $\widehat{a}_{\boldsymbol{v},m}$ is a representative of the inverse of $a_{\boldsymbol{v},m}$ modulo g (chosen arbitrarily). We will initialize the oracle $\mathcal{M}$ with these elements $(a_{\boldsymbol{v},i}, \boldsymbol{v})$, for $\boldsymbol{v} \in \mathcal{A}$.

With the notations above, we have that for all $\boldsymbol{v} \in \mathcal{A}$ of weight 1,

$$H(a_{\boldsymbol{v},1}, a_{\tilde{\boldsymbol{v}},1}, \ldots, a_{\boldsymbol{v},m}, a_{\tilde{\boldsymbol{v}},m}) = \sum_{i=0}^{m} a_{\boldsymbol{v},i} a_{\tilde{\boldsymbol{v}},i} = (r'_{\boldsymbol{v}} + a_{\boldsymbol{v},m} r_{\boldsymbol{v}}) g$$

for some $r'_{\boldsymbol{v}}$ that depends on the $a_{\boldsymbol{v},i}$'s and $a_{\tilde{v},j}$'s (with $i \leq m$ and $j \leq m-1$) but not on $r_{\boldsymbol{v}}$. Hence, as $r_{\boldsymbol{v}}$ has min-entropy at least n and R is an integral domain, we have

$$H(a_{\boldsymbol{v},1}, a_{\tilde{\boldsymbol{v}},1}, \ldots, a_{\boldsymbol{v},m}, a_{\tilde{\boldsymbol{v}},m}) = \tilde{r}_{\boldsymbol{v}} g \tag{A.1}$$

for some $\tilde{r}_{\boldsymbol{v}}$ with min-entropy at least n. Moreover, knowing the $\tilde{r}_{\boldsymbol{w}}$ for $\boldsymbol{w} \neq \boldsymbol{v}$ ($\boldsymbol{w}$ of weight 1) does not decrease this min-entropy (because $r_{\boldsymbol{v}}$ is independent of the $\tilde{r}_{\boldsymbol{w}}$). Hence, we have that

$$p_{\max}(\{\tilde{r}_{\boldsymbol{v}}\}_{\boldsymbol{v} \in \mathcal{A} \text{ of weight } 1}) \leq 2^{-n}. \tag{A.2}$$

Theorem A.4. *Assume we initialize the oracle $\mathcal{M}$ with the pairs $(a_{\boldsymbol{v},i}, \boldsymbol{v})$ defined above, for $\boldsymbol{v} \in \mathcal{A}$ and $1 \leq i \leq m$. Then, for any probabilistic polynomial time adversary $\mathfrak{A}$ interacting with the oracle $\mathcal{M}$, the probability that $\mathfrak{A}$ manages to make $\mathcal{M}$ output "WIN" is negligible in n. In other words, our simple setting of the GGH multilinear map is secure in the weak multilinear map model (see Definition A.1).*

We remark that the conditions on the distributions D_a and D_r are satisfied by all the sampling methods described in Chapter 5. Hence, the theorem proves that our simple setting of the GGH multilinear map is secure in the weak multilinear map model, independently of the sampling method chosen (among the ones described in this article).

Proof. For simplicity of notation, we will sometimes index the elements of $\mathcal{A}$ by $\boldsymbol{v_1}, \cdots, \boldsymbol{v_{2\kappa}}$.

In this proof, we will merge the arithmetic queries on the encodings and the zero-testing queries by saying that the adversary $\mathfrak{A}$ directly sends a polynomial p to the oracle. Then $\mathcal{M}$ performs the arithmetic operations on the encodings that correspond to the polynomial p (if they are relevant) and applies the zero-testing procedure on the result.

The idea of the proof is to show that the only encodings that $\mathfrak{A}$ can query that will pass the zero-testing procedure are linear combinations of elements of the form $H(a_{\boldsymbol{v},1}, a_{\tilde{\boldsymbol{v}},1}, \ldots, a_{\boldsymbol{v},m}, a_{\tilde{\boldsymbol{v}},m})$ (all other polynomials in the $a_{\boldsymbol{v},i}$'s will fail to pass the zero test with high probability). Then, each zero-test on a $H(a_{\boldsymbol{v},1}, a_{\tilde{\boldsymbol{v}},1}, \ldots, a_{\boldsymbol{v},m}, a_{\tilde{\boldsymbol{v}},m})$ will result in a handle of a random element (because of the randomness contained in $r_{\boldsymbol{v}}$), and all these elements will be independent. Hence the adversary has negligible probability of finding a polynomial that annihilates them.

[2] In the GGH multilinear map, the distribution D_a should be a Gaussian distribution (whose shape depends on the sampling method). This has no importance for our proof, so we make no assumption about it here.

Lemma A.5. *Let P be a polynomial in the variables $(X_{\boldsymbol{v},i})_{\{\boldsymbol{v}\in\mathcal{A},1\leq i\leq m\}}$ generated by the attacker $\mathfrak{A}$ such that $P(a_{\boldsymbol{v_1},1},\cdots,a_{\boldsymbol{v_1},m},a_{\boldsymbol{v_2},1},\cdots,a_{\boldsymbol{a_{2\kappa}},m})=0 \mod g$. Then, with overwhelming probability, we have*

$$P(X_{\boldsymbol{v_1},1},\cdots,X_{\boldsymbol{v_1},m},X_{\boldsymbol{v_2},1},\cdots,X_{\boldsymbol{v_{2\kappa}},m}) = \sum_{\substack{\boldsymbol{v}\in\mathcal{A}\\ \|\boldsymbol{v}\|_1=1}} \alpha_{\boldsymbol{v}} H(X_{\boldsymbol{v},1},X_{\tilde{\boldsymbol{v}},1},\cdots,X_{\boldsymbol{v},m},X_{\tilde{\boldsymbol{v}},m})$$

for some scalars $\alpha_{\boldsymbol{v}}\in R$.

This lemma means that the only encodings that will pass the zero test that the attacker can create are the $H(u_{\boldsymbol{v},1},u_{\tilde{\boldsymbol{v}},1},\cdots,u_{\boldsymbol{v},m},u_{\tilde{\boldsymbol{v}},m})$ and linear combination of them. As zero-testing linear combinations of encodings that pass the zero test does not provide more information that what was revealed by zero-testing the original encodings, we will assume in the following that the adversary makes zero testing queries for $H(u_{\boldsymbol{v},1},u_{\tilde{\boldsymbol{v}},1},\cdots,u_{\boldsymbol{v},m},u_{\tilde{\boldsymbol{v}},m})$ for all $\boldsymbol{v}\in\mathcal{A}$ of weight 1, and that they are the only queries that pass the zero test.

Recall that the numerator of $H(u_{\boldsymbol{v},1},u_{\tilde{\boldsymbol{v}},1},\cdots,u_{\boldsymbol{v},m},u_{\tilde{\boldsymbol{v}},m})$ is of the form $H(a_{\boldsymbol{v},1},a_{\tilde{\boldsymbol{v}},1},\cdots,a_{\boldsymbol{v},m},a_{\tilde{\boldsymbol{v}},m}) = \tilde{r}_{\boldsymbol{v}}g$ (see Equality A.1). Hence, after all its zero-test queries, the attacker $\mathfrak{A}$ gets handles to the values $\tilde{r}_{\boldsymbol{v}}$ for all $\boldsymbol{v}\in\mathcal{A}$ of weight 1. These are the only post-zero-test handles that the attacker obtains. These handles map to random elements $\tilde{r}_{\boldsymbol{v}}$ that may not be independent, but that have a lot of (conditioned) min-entropy. Hence, it is very unlikely that the attacker creates a non zero polynomial that annihilates these random values. More formally, let P be a polynomial of degree $d=\text{poly}(n)$ queried by the attacker in the post-zero-test phase. Then, using the improved Schwartz-Zippel lemma of [MSZ16] (Lemma A.3) in K, we have that

$$\Pr[P(\{\tilde{r}_{\boldsymbol{v}}\}_{\boldsymbol{v}\in\mathcal{A}\text{ of weight }1})=0]\leq d\cdot 2^{-n}=\text{negl}(n)$$

using the fact that $p_{\max}(\{\tilde{r}_{\boldsymbol{v}}\}_{\boldsymbol{v}\in\mathcal{A}\text{ of weight }1})\leq 2^{-n}$ (see Inequality (A.2)).

Hence, the attacker has negligible probability of creating a non zero polynomial P, of degree polynomial in n, that annihilates the post-zero-test handles. This concludes the proof of our theorem. □

Proof of Lemma A.5. **Step 1.** First, let $\boldsymbol{v}\in\mathcal{A}$ be of weight 1 and let P be a polynomial in the variables $\{X_{\boldsymbol{v},1},X_{\tilde{\boldsymbol{v}},1},\cdots,X_{\boldsymbol{v},m},X_{\tilde{\boldsymbol{v}},m}\}$ (and not in all $X_{\boldsymbol{w},i}$ for $\boldsymbol{w}\in\mathcal{A}$ and $i\leq m$) such that

$$P(a_{\boldsymbol{v},1},a_{\tilde{\boldsymbol{v}},1},\cdots,a_{\boldsymbol{v},m},a_{\tilde{\boldsymbol{v}},m})=0 \mod g.$$

We will show that $P=\alpha H \mod g$ for some $\alpha\in R/gR$. Note that as we are only interested (for the moment) in equalities modulo g, we will assume that our polynomial P has coefficients in R/gR (which is a field as gR is a prime ideal). We will also see the $a_{\boldsymbol{v},i}$ as elements of R/gR.

Using the fact that the polynomial $P(u_{\boldsymbol{v},1},u_{\tilde{\boldsymbol{v}},1},\cdots,u_{\boldsymbol{v},m},u_{\tilde{\boldsymbol{v}},m})$ is a valid encoding at level $\boldsymbol{v}^*$, we know that

$$P=P_1(X_{\boldsymbol{v},1},X_{\tilde{\boldsymbol{v}},1},\cdots,X_{\boldsymbol{v},m})+X_{\tilde{\boldsymbol{v}},m}P_2(X_{\boldsymbol{v},1},X_{\boldsymbol{v},2},\cdots,X_{\boldsymbol{v},m})$$

for some polynomials P_1 of degree 2 and P_2 of degree 1. We cannot apply the Schwartz-Zippel lemma to P because the variable $a_{\tilde{\boldsymbol{v}},m} \mod g$ is completely determined by the other variables that appears in the polynomial. So we first introduce a new polynomial that does not depend on $X_{\tilde{\boldsymbol{v}},m}$ before applying the Schwartz-Zippel lemma.

We define the polynomial $Q\in(R/gR)[X_{\boldsymbol{v},1},X_{\tilde{\boldsymbol{v}},1},\cdots,X_{\boldsymbol{v},m}]$ (note that $X_{\tilde{\boldsymbol{v}},m}$ does not appear in Q) by

$$\begin{aligned}Q(X_{\boldsymbol{v},1},X_{\tilde{\boldsymbol{v}},1},\cdots,X_{\boldsymbol{v},m}) =& X_{\boldsymbol{v},m}P_1(X_{\boldsymbol{v},1},X_{\tilde{\boldsymbol{v}},1},\cdots,X_{\boldsymbol{v},m})\\ &-\left(\sum_{i=1}^{m-1}X_{\boldsymbol{v},i}X_{\tilde{\boldsymbol{v}},i}\right)P_2(X_{\boldsymbol{v},1},X_{\boldsymbol{v},2},\cdots,X_{\boldsymbol{v},m}).\end{aligned}$$

Using the fact that $a_{\tilde{\boldsymbol{v}},m}=-a_{\boldsymbol{v},m}^{-1}\sum_{i=1}^{m-1}a_{\boldsymbol{v},i}a_{\tilde{\boldsymbol{v}},i} \mod g$, we have that

$$Q(a_{\boldsymbol{v},1},a_{\tilde{\boldsymbol{v}},1},\cdots,a_{\boldsymbol{v},m})=a_{\boldsymbol{v},m}P(a_{\boldsymbol{v},1},a_{\tilde{\boldsymbol{v}},1},\cdots,a_{\boldsymbol{v},m},a_{\tilde{\boldsymbol{v}},m})=0 \mod g.$$

But the variables $a_{\boldsymbol{v},1},a_{\tilde{\boldsymbol{v}},1},\cdots,a_{\boldsymbol{v},m}$ are drawn from D_a independently with guessing probability at most 2^{-n} (even when reduced modulo g), and the degree of Q is at most 3. So using Schwartz-Zippel

lemma (Lemma A.3) in R/gR, we have that Q should be the zero polynomial, except with negligible probability. In the following, we will then assume that $Q = 0$. This means that we have the equality between polynomials $P_1 = X_{\boldsymbol{v},m}^{-1} \left(\sum_{i=1}^{m-1} X_{\boldsymbol{v},i} X_{\tilde{\boldsymbol{v}},i}\right) P_2$. Hence, we can re-write

$$\begin{aligned} P &= P_2 \left(X_{\boldsymbol{v},m}^{-1} \left(\sum_{i=1}^{m-1} X_{\boldsymbol{v},i} X_{\tilde{\boldsymbol{v}},i} \right) + X_{\tilde{\boldsymbol{v}},m} \right) \\ &= P_2 X_{\boldsymbol{v},m}^{-1} \cdot H(X_{\boldsymbol{v},1}, X_{\tilde{\boldsymbol{v}},1}, \cdots, X_{\boldsymbol{v},m}, X_{\tilde{\boldsymbol{v}},m}). \end{aligned}$$

As we know that P is a polynomial and $X_{\boldsymbol{v},m}$ does not divide H, this means that it divides P_2. However, P_2 is of degree 1, hence we conclude that $P = \alpha H \mod g$ for some scalar $\alpha \in R$.

Step 2. Now, let P be a polynomial in all the variables $X_{\boldsymbol{v},i}$ for $\boldsymbol{v} \in \mathcal{A}$ and $i \leq m$ such that $P((a_{\boldsymbol{v},i})_{\boldsymbol{v}\in\mathcal{A}, i\leq m}) = 0 \mod g$. We will prove by induction on κ (recall that κ is the number of $\boldsymbol{v} \in \mathcal{A}$ of weight 1) that

$$P(X_{\boldsymbol{v_1},1}, \cdots, X_{\boldsymbol{v_1},m}, X_{\boldsymbol{v_2},1}, \cdots, X_{\boldsymbol{v_{2\kappa}},m}) = \sum_{\substack{\boldsymbol{v}\in\mathcal{A} \\ \text{of weight 1}}} \alpha_{\boldsymbol{v}} H(X_{\boldsymbol{v},1}, X_{\tilde{\boldsymbol{v}},1}, \cdots, X_{\boldsymbol{v},m}, X_{\tilde{\boldsymbol{v}},m}) \mod g.$$

The case $\kappa = 1$ was already done above (this is exactly step 1). Assume then that $\kappa > 1$. Let $\boldsymbol{v}_1 \in \mathcal{A}$ be of weight 1 and assume without loss of generality that $\boldsymbol{v}_2 = \tilde{\boldsymbol{v}}_{\boldsymbol{1}}$. We define the polynomial $\tilde{P}$ in the variables $\{X_{\boldsymbol{v}_1,i}, X_{\boldsymbol{v}_2,j}\}_{i,j\leq m}$ to be the polynomial P where $X_{\boldsymbol{v}_j,i}$ is evaluated at $a_{\boldsymbol{v}_j,i}$ for all $j \geq 3$ and $i \leq m$, i.e.,

$$\tilde{P}(X_{\boldsymbol{v}_1,1}, X_{\boldsymbol{v}_1,2}, \cdots, X_{\boldsymbol{v}_2,m}) = P(X_{\boldsymbol{v}_1,1}, \cdots, X_{\boldsymbol{v}_2,m}, a_{\boldsymbol{v}_3,1}, \cdots, a_{\boldsymbol{v}_{2\kappa},m}).$$

By hypothesis, we have that $\tilde{P}(a_{\boldsymbol{v}_1,1}, a_{\boldsymbol{v}_1,1}, \cdots, a_{\boldsymbol{v}_2,m}) = 0 \mod g$. Now, by step 1, this means that $\tilde{P} = \alpha H \mod g$ for some $\alpha \in R$. Using the structure of the levels of the encodings, we then know that

$$\begin{aligned} P(X_{\boldsymbol{v}_1,1}, \cdots, X_{\boldsymbol{v}_{2\kappa},m}) =& \alpha H(X_{\boldsymbol{v}_1,1}, \cdots, X_{\boldsymbol{v}_2,m}) + \sum_{\boldsymbol{i}=1}^{m} P_i(X_{\boldsymbol{v}_3,1}, \cdots, X_{\boldsymbol{v}_{2\kappa},m}) X_{\boldsymbol{v}_1,i} \\ &+ T(X_{\boldsymbol{v}_3,1}, \cdots, X_{\boldsymbol{v}_{2\kappa},m}) \end{aligned}$$

for some polynomials P_i and T in R/gR such that $P_i(a_{\boldsymbol{v}_3,1}, \cdots, a_{\boldsymbol{v}_{2\kappa},m}) = 0$ and $T(a_{\boldsymbol{v}_3,1}, \cdots, a_{\boldsymbol{v}_{2\kappa},m}) = 0$. By induction hypothesis, we then know that the polynomials P_i and T are linear combinations of the polynomial H evaluated at different $X_{\boldsymbol{v},i}$. This implies that if P_i is non zero, then $P_i(u_{\boldsymbol{v}_3,1}, \cdots, u_{\boldsymbol{v}_{2\kappa},m}) u_{\boldsymbol{v}_1,i}$ is an encoding at level $\boldsymbol{v}^* + v_1$ which is not an admissible level. Hence, we have that $P_i = 0$ for all i and, by induction hypothesis on T, we obtain the desired result.

Step 3. We have proven that for all polynomial P that the adversary can query, which passes the zero test, then with overwhelming probability we have

$$P(X_{\boldsymbol{v_1},1}, \cdots, X_{\boldsymbol{v_{2\kappa}},m}) = \sum_{\substack{\boldsymbol{v}\in\mathcal{A} \\ \|\boldsymbol{v}\|_1=1}} \alpha_{\boldsymbol{v}} H(X_{\boldsymbol{v},1}, X_{\tilde{\boldsymbol{v}},1}, \cdots, X_{\boldsymbol{v},m}, X_{\tilde{\boldsymbol{v}},m}) + gT$$

for some polynomial T in R. It remains to show that $T = 0$, except with negligible probability. We observe that if the attacker knows a polynomial of the above form with $T \neq 0$, then it can recover a multiple of g. Indeed, in $\sum_{\substack{\boldsymbol{v}\in\mathcal{A} \\ \|\boldsymbol{v}\|_1=1}} \alpha_{\boldsymbol{v}} H(X_{\boldsymbol{v},1}, X_{\tilde{\boldsymbol{v}},1}, \cdots, X_{\boldsymbol{v},m}, X_{\tilde{\boldsymbol{v}},m})$, all monomials $X_{\boldsymbol{v},i} X_{\tilde{\boldsymbol{v}},i}$ have the same coefficient $\alpha_{\boldsymbol{v}}$ when i varies and $\boldsymbol{v}$ is fixed. This means that we can recover a multiple of g by computing the difference of two such coefficients in our polynomial $\sum_{\substack{\boldsymbol{v}\in\mathcal{A} \\ \|\boldsymbol{v}\|_1=1}} \alpha_{\boldsymbol{v}} H(X_{\boldsymbol{v},1}, X_{\tilde{\boldsymbol{v}},1}, \cdots, X_{\boldsymbol{v},m}, X_{\tilde{\boldsymbol{v}},m}) + gT$ (at least one should be non zero if $T \neq 0$). Hence, if the adversary queries such a polynomial, it knows a multiple of g. However, the handles output by the oracle reveal nothing about g and g is chosen with sufficiently high min-entropy, hence we will show that the adversary cannot create a multiple of g except with negligible probability.

The idea is that if the attacker only performs zero-test queries that do not pass the zero-test or queries for polynomials of the form $\sum_{\substack{\boldsymbol{v}\in\mathcal{A} \\ \|\boldsymbol{v}\|_1=1}} \alpha_{\boldsymbol{v}} H(X_{\boldsymbol{v},1}, X_{\tilde{\boldsymbol{v}},1}, \cdots, X_{\boldsymbol{v},m}, X_{\tilde{\boldsymbol{v}},m})$ (without the term gT),

then it does not learn enough information to obtain a multiple of g with non negligible probability. Hence, it cannot ask for a polynomial of the form $\sum_{\|\boldsymbol{v}\|_1=1} \alpha_{\boldsymbol{v}} H(X_{\boldsymbol{v},1}, X_{\tilde{\boldsymbol{v}},1}, \cdots, X_{\boldsymbol{v},m}, X_{\tilde{\boldsymbol{v}},m}) + gT$ with $T \neq 0$. Assume then that the attacker only queries polynomials of the form $\sum_{\|\boldsymbol{v}\|_1=1} \alpha_{\boldsymbol{v}} H(X_{\boldsymbol{v},1}, X_{\tilde{\boldsymbol{v}},1}, \cdots, X_{\boldsymbol{v},m}, X_{\tilde{\boldsymbol{v}},m})$ and polynomials that do not pass the zero test. We prove that all other possible choices of g, except a negligible fraction of them, would have led to the same answers of the oracle. Hence, the adversary cannot guess the value of g except with negligible probability.

The queries on polynomials of the form $\sum_{\|\boldsymbol{v}\|_1=1} \alpha_{\boldsymbol{v}} H(X_{\boldsymbol{v},1}, X_{\tilde{\boldsymbol{v}},1}, \cdots, X_{\tilde{\boldsymbol{v}},m})$ do not leak any information because for all values of g, we know that they should pass the zero test (and the handle that is generated when a query passes the zero test is independent of the choice of g). The queries on polynomials that do not pass the zero test may leak some information, but this leakage will be negligible compared to the entropy of g. Let $c_1, \cdots, c_k$ be the numerators of all encodings that the adversary queried and that did not pass the zero test. Then, any g that is not a divisor of $c_1 c_2 \cdots c_k$ would have given the same answers. So the number of "bad" choices of g is at most the number of divisors of $c_1 c_2 \cdots c_k$. Denoting by $\mathcal{N}$ the algebraic norm of elements in K, we have that $\mathcal{N}(x) > 2$ if $x \in R$ is non invertible, and if $x|y$, then $\mathcal{N}(x)|\mathcal{N}(y)$. Hence, the number of non invertible divisors of $c_1 c_2 \cdots c_k$ is at most $\log_2(\mathcal{N}(c_1 c_2 \cdots c_k))$. This is polynomial in n. Indeed, the element $c_1 c_2 \cdots c_k$ was computed by a polynomial-time attacker, so its coefficients are bounded by 2^{n^c} for some constant c (the attacker has to write this element with a polynomial number of bits). To conclude, there are only a polynomial number of "bad" g (i.e., a polynomial number of non invertible elements that divide $c_1 c_2 \cdots c_k$). On the contrary, there is an exponential number of possible g when we generate the parameters of our multilinear map. Hence, the attacker has negligible chance to be able to guess a multiple of g if it knows nothing about it. This achieves the proof of our Lemma A.5. □

Appendix B

Adapting the quantum attack to circuit obfuscators

In this section, we explain how the quantum attack of Section 6.2 can be extended to the circuit obfuscators of [AB15, Zim15, BD16, DGG+18], when instantiated with the GGH13 map. The structure of these obfuscators is very different from the abstract obfuscator described in Section 6.2 and so the attack described in Section 6.3 cannot be directly applied to it. However, similarly to the other obfuscators described above, the circuit obfuscators also use the levels of the GGH13 multilinear map to prevent mixed-input attacks. This is the weakness we exploited to mount a mixed-input attack against the abstract obfuscator, and here again, this will enable us to attack these circuit obfuscators. In this section, we first describe a simplified circuit obfuscator, which captures the obfuscators of [AB15,Zim15,BD16,DGG+18]. We then show how to adapt our attack to mount a quantum polynomial-time mixed-input attack against this simple circuit obfuscator.

B.1 The simple circuit obfuscator

The circuit obfuscators can be instantiated with the GGH13 multilinear map [GGH13a] in its asymmetric version, but with a composite g. More concretely, we sample three elements $g_1, g_2, g_3 \in R$ as for the original g in the GGH13 map, that is $\|g_i\| = O(n)$, $\|1/g_i\| = O(n^2)$ and such that $\mathcal{N}(g_i)$ is a prime integer, for all $i \in [3]$. Then, we let $g = g_1 g_2 g_3$. If we denote by $R_i = R/g_i R$ the quotient rings for $i \in [3]$, then using the Chinese remainder theorem we know that the encoding space R/gR is isomorphic to $R_1 \times R_2 \times R_3$. In the following, it will be useful to choose this point of view, as we will encode triplets of elements $(a_1, a_2, a_3) \in R_1 \times R_2 \times R_3$, using the GGH13 map.

Let Σ be some subset of $\{0,1\}^l$ with both l and $|\Sigma|$ that are polynomial in the security parameter λ. We will be interested in arithmetic circuits $C : \Sigma \to \{0,1\}$. By arithmetic circuits, we mean that C performs addition, multiplication and subtraction over the bits of the elements of Σ (i.e., C is an arithmetic circuit from $\{0,1\}^l$ to $\{0,1\}$, but we are only interested in its restriction to $\Sigma \subseteq \{0,1\}^l$). The operations over the bits are performed over $\mathbb{Z}$ but we only consider circuits whose output is in $\{0,1\}$. Let $\mathcal{C}$ be a class of such circuits, whose size is bounded by some polynomial (the properties of this class of circuit will not be interesting for our attack) and let U be a universal circuit for the class $\mathcal{C}$. The size of U is also bounded by some polynomial in the security parameter. We abuse notation by denoting by C both a circuit of $\mathcal{C}$ and its bit representation, that is we have $U(\sigma, C) = C(\sigma)$ for any $\sigma \in \Sigma$ (the first C denotes the bit representation of the circuit while the second one represents the function computed by the circuit).

To obfuscate a circuit C of the class $\mathcal{C}$, the main idea of the circuit obfuscator is to produce GGH13 encodings of the bits of C and of the bits of all the possible inputs $\sigma \in \Sigma$. Then, to evaluate the obfuscated circuit, it suffices to homomorphically evaluate the universal circuit U on these encodings and to test whether the result is 0 or not. In order to prove the security of their obfuscators, the authors of [AB15, Zim15, BD16, DGG+18] added other gadgets to their obfuscators. The first idea is to encode the useful information only in the second slot of the GGH13 map (in the ring R_2) and to use the two other slots to prevent some mixed-input attack (where we mix the bits of two circuits). They also use straddling set systems, like the abstract obfuscator defined in Section 6.2, to prevent other kinds of mixed input attacks (where we mix the bits of two inputs). We describe below in more detail how the

circuit obfuscators proceed, given as input a circuit $C \in \mathcal{C}$. In order to help understanding what is happening, we also describe in parallel how to evaluate the obfuscated circuit.

1. First, we encode each bit of all the possible inputs $\sigma \in \Sigma$ (recall that we chose $|\Sigma|$ to be polynomial in the security parameter, so it is possible to enumerate all the elements of Σ). For each symbol $\sigma \in \Sigma$ and each bit position $i \in [l]$, define $W_{i,\sigma}^{(1)} = [r_\sigma^{(1)} \cdot w_{i,\sigma}^{(1)}]_{\boldsymbol{v}_\sigma^{(1)}}$ and $R_\sigma^{(1)} = [r_\sigma^{(1)}]_{\boldsymbol{v}_\sigma^{(1)}}$, where $r_\sigma^{(1)}$ is sampled uniformly in $R/gR^\times$ (and only depends on σ) and

$$w_{i,\sigma}^{(1)} = (y_i^{(1)}, \sigma_i, \rho_{i,\sigma}^{(1)}) \in R_1 \times R_2 \times R_3,$$

   for σ_i the i-th bit of σ and $y_i^{(1)}$ and $\rho_{i,\sigma}^{(1)}$ sampled uniformly in R_1 and R_3 respectively. The level $\boldsymbol{v}_\sigma^{(1)}$ of the encoding will be chosen to prevent mixed-input attacks. We will go into more details about the levels of the encodings later. These encodings $W_{i,\sigma}^{(1)}$ and $R_\sigma^{(1)}$ are made public, for $i \in [l]$ and $\sigma \in \Sigma$. Note that $y_i^{(1)}$ is the same for all symbols σ, this will be necessary for correctness.

2. Second, we encode the bits of the representation of the circuit $C \in \mathcal{C}$. We denote by $|C|$ the size of the bit representation of C. For each $1 \leq j \leq |C|$, define $W_j^{(2)} = [r^{(2)} \cdot w_j^{(2)}]_{\boldsymbol{v}^{(2)}}$ and $R^{(2)} = [r^{(2)}]_{\boldsymbol{v}^{(2)}}$, where $r^{(2)}$ is sampled uniformly in $R/gR^\times$ and

$$w_j^{(2)} = (y_j^{(2)}, C_j, \rho_j^{(2)}) \in R_1 \times R_2 \times R_3,$$

   for C_j the j-th bit of the representation of C and $y_j^{(2)}$ and $\rho_j^{(2)}$ sampled uniformly in R_1 and R_3 respectively. Again, the level $\boldsymbol{v}^{(2)}$ of the encoding will be described later. These encodings $W_j^{(2)}$ and $R^{(2)}$ are made public, for $1 \leq j \leq |C|$.

Once we have encodings for the bits of C and for all the possible input values $\sigma \in \Sigma$, as the universal circuit U only performs additions, subtractions and multiplications, we can homomorphically evaluate it on the encodings. We can always perform multiplications of encodings, it will only increase the level of the encodings. However, there is a subtlety for addition and subtraction, as we can only add and subtract encodings at the same level. To circumvent this difficulty, one can use the encodings $R^{(2)}$ and $R_\sigma^{(1)}$. During the evaluation of the universal circuit U on the encodings, we will perform computations so that for all intermediate encodings we compute, we always have encodings of the form $[r \cdot w]_{\boldsymbol{v}}$ and $[r]_{\boldsymbol{v}}$, with the same level $\boldsymbol{v}$. At the beginning, all the encodings described above have the desired form $[r \cdot w]_{\boldsymbol{v}}$ and $[r]_{\boldsymbol{v}}$. If we want to multiply $[r_1 \cdot w_1]_{\boldsymbol{v}_1}$ and $[r_2 \cdot w_2]_{\boldsymbol{v}_2}$, we just compute the product of the encodings to get $[r_1 r_2 \cdot w_1 w_2]_{\boldsymbol{v}_1+\boldsymbol{v}_2}$ and we also compute the product of the r part to obtain $[r_1 r_2]_{\boldsymbol{v}_1+\boldsymbol{v}_2}$. Note that here, the levels $\boldsymbol{v}_1 + \boldsymbol{v}_2$ might have coefficients that are larger than 1. If we want to add $[r_1 \cdot w_1]_{\boldsymbol{v}_1}$ and $[r_2 \cdot w_2]_{\boldsymbol{v}_2}$, then two cases appear. If $r_1 = r_2$ and $\boldsymbol{v}_1 = \boldsymbol{v}_2$, then we add both encodings to get $[r_1 \cdot (w_1 + w_2)]_{\boldsymbol{v}_1}$ and keep $[r_1]_{\boldsymbol{v}_1}$. Otherwise, we compute $[r_1]_{\boldsymbol{v}_1} \cdot [r_2 \cdot w_2]_{\boldsymbol{v}_2} + [r_2]_{\boldsymbol{v}_2} \cdot [r_1 \cdot w_1]_{\boldsymbol{v}_1} = [r_1 r_2 \cdot (w_1 + w_2)]_{\boldsymbol{v}_1+\boldsymbol{v}_2}$ and compute the product $[r_1 r_2]_{\boldsymbol{v}_1+\boldsymbol{v}_2}$. We proceed similarly for subtraction.

With this technique, we can evaluate the circuit U on the encodings provided by the obfuscator, independently of the levels used to encode them. Assume we evaluate it honestly on the encodings of C and of some input $\sigma \in \Sigma$, we then obtain encodings $W_\sigma = [r_\sigma \cdot w_\sigma]_{\boldsymbol{v}_\sigma}$ and $R_\sigma = [r_\sigma]_{\boldsymbol{v}_\sigma}$ at some level $\boldsymbol{v}_\sigma$, for some $r_\sigma \in R/gR$, where

$$w_\sigma = (y^*, C(\sigma), \rho_\sigma) \in R_1 \times R_2 \times R_3,$$

for some $y^* \in R_1$ and $\rho_\sigma \in R_3$. Note that, as the $y_i^{(1)}$'s do not depend on the input σ, the value y^* is the same for all σ's. We then want to annihilate the values in the extra slots (that is y^* and ρ_σ) to recover the value of $C(\sigma)$ by zero-testing. To do that, the obfuscator provides two more encodings.

3. To annihilate the value in the third slot, the obfuscator output encodings $\widehat{W_\sigma} = [\widehat{r_\sigma} \cdot \widehat{w}]_{\widehat{\boldsymbol{v}_\sigma}}$ and $\widehat{R_\sigma} = [\widehat{r_\sigma}]_{\widehat{\boldsymbol{v}_\sigma}}$, for all $\sigma \in \Sigma$, where $\widehat{r_\sigma}$ is sampled uniformly in $R/gR^\times$ and

$$\widehat{w} = (\widehat{y}, \widehat{\alpha}, 0),$$

   for $\widehat{y}$ and $\widehat{\alpha}$ uniformly chosen in R_1 and $R_2^\times$, respectively.

Multiplying the encoding of $w_\sigma = (y^*, C(\sigma), \rho_\sigma)$ obtained above, by this encoding of $\widehat{w} = (\widehat{y}, \widehat{\alpha}, 0)$ enables us to cancel the last slot and to obtain an encoding of $\widehat{w}_\sigma := (\widehat{y} \cdot y^*, \widehat{\alpha} \cdot C(\sigma), 0)$. We also multiply the r parts, as described above. Note that to cancel this third slot, the obfuscator outputs one pair of encodings for each symbol $\sigma \in \Sigma$. While this may seem useless because each encoding encodes the same $\widehat{w}$, this is in fact required to standardise the levels of the encodings. Indeed, after evaluating the universal circuit on the encodings of C and σ, we obtain an encoding whose level depends on σ. By multiplying with an encoding at a complementary level at this step, we can then ensure that the level of the product is independent of σ. This property will be important, because to zero-test the final encoding, we need it to be at the maximal level $\boldsymbol{v}^*$, independently of the input σ.

4. Finally, to cancel the first slot, the obfuscator provides two encodings $\overline{W} = [\overline{r} \cdot \overline{w}]_{\overline{\boldsymbol{v}}}$ and $\overline{R} = [\overline{r}]_{\overline{\boldsymbol{v}}}$, where $\overline{r}$ is sampled uniformly in $R/gR^\times$ and

$$\overline{w} = (\widehat{y} \cdot y^*, 0, 0).$$

Note that $\widehat{w}_\sigma - \overline{w} = 0$ if and only if $C(\sigma) = 0$. Hence, it suffices to subtract the corresponding encodings (using the r part, because the levels of the encodings will not match) and to zero-test the obtained encoding to determine whether $C(\sigma) = 0$ or 1.

This completes the description of the simple circuit obfuscator, together with the correctness proof of the evaluation of the obfuscated program. Before describing the mixed-input attack, we would like to insist on some properties of the obfuscator described above.

- The levels of the encodings output by the obfuscator are chosen such that all honest evaluations of the obfuscated circuit on some input $\sigma \in \Sigma$ produce encodings with the same level. This level is then chosen to be the maximal level of the GGH13 map, and will be denoted by $\boldsymbol{v}^*$. The obfuscator also provides a zero-test parameter p_{zt} to enable zero-test at level $\boldsymbol{v}^*$. In the following, the only thing that will be interesting for our attack is that a honest evaluation of the obfuscated circuit on any input $\sigma \in \Sigma$ outputs an encoding at level $\boldsymbol{v}^*$, so we do not go into more details about the levels of the encodings.

- As we already noted, the value y^* obtained in the first slot after evaluating the universal circuit on the encodings of C and σ does not depend on σ. This is needed for the last step, where we subtract $\widehat{y} \cdot y^*$. As we want this to output 0 for any input (to cancel out the first slot), the value y^* has to be independent of σ. This first slot prevents us from mixing the bits of the circuit C, but does not prevent us from mixing the bits of the input σ (i.e., changing the value of some bit during the evaluation). Mixing the bits of the input is only prevented by the GGH13 map and the straddling set system (recall that the levels of the encodings depend on the input σ). This is the kind on mixed-input attack we will be able to perform after recovering the secret element h of the GGH13 map.

We observe that the circuit obfuscator described above is a simplification of the circuit obfuscators of [AB15,Zim15,BD16,DGG+18]. However, it captures the main techniques used by these constructions, and the attack we describe below is not impacted by the small changes between this simple obfuscator and the actual constructions.

B.2 The mixed-input attack

As mentioned above, the attack will consist in modifying a bit of the input σ during the computation. The idea is the same as for the attack of Section 6.3. As previously, we start by constructing a new zero-testing parameter p'_{zt} at level $2\boldsymbol{v}^*$ (testing whether the numerator of an encoding is a multiple of g^2, and not only g). This step can be performed exactly as in Section 6.3.1 and we do not re-explain it here. Recall that to perform this step, we need to be able to evaluate the obfuscated circuit on inputs for which the output is zero.

The second part of the attack (using p'_{zt} to mount a mixed input attack) will differ from the one for the abstract branching program obfuscator. The first difference is that in the abstract branching program obfuscator, we only computed products of matrices. So by changing a matrix, we just changed

the final level of the encodings but all the operations remained possible (products of encodings are always possible, whatever their levels are). Here, as we evaluate a circuit with additions and multiplications, we must be careful. Indeed, if we change the level of one encoding of a sum but not the other one, we will not be able to perform the sum anymore. To circumvent this difficulty, we will use a specific universal circuit, which terminates with a multiplication. Let U be a universal circuit for the class of circuit $\mathcal{C}$. We define a new circuit $\tilde{U}$, which takes as input a concatenation of the description of two circuits in $\mathcal{C}$ and an input $\sigma \in \Sigma$ and computes the product of the evaluations of the two circuits on input σ. More formally, we define

$$\tilde{U}(\sigma, C_1 \cdot C_2) = U(\sigma, C_1) \cdot U(\sigma, C_2).$$

The circuit $\tilde{U}$ is a universal circuit for the class $\mathcal{C} \cdot \mathcal{C}$. Note that when evaluating the circuit $\tilde{U}$, we finish the evaluation with a multiplication. To perform our mixed input attack, we will evaluate $U(\cdot, C_1)$ and $U(\cdot, C_2)$ honestly on different inputs σ_1 and σ_2. As each partial evaluation is honest, we can perform all the required operations on the encodings. The dishonest computation will be the last multiplication only.

Let σ_1 and σ_2 be two distinct elements of Σ. Let C_{00} be a circuit that evaluates always to 0 on Σ. We also let C_{10} be a circuit that evaluates to 1 on σ_1 and to 0 otherwise and C_{01} be a circuit that evaluates to 1 on σ_2 and to 0 otherwise. The functions computed by $C_{00} \cdot C_{00}$ and by $C_{01} \cdot C_{10}$ are the same, so these circuits are equivalent. We will now show how to distinguish the obfuscated versions of $C_{00} \cdot C_{00}$ and $C_{01} \cdot C_{10}$, when using the universal circuit $\tilde{U}$. As both circuits are equivalent, this will result into an attack against the iO security of the obfuscator.

Objective: The obfuscator obfuscates the circuit $C_1 \cdot C_2 \in \{C_{00} \cdot C_{00}, C_{01} \cdot C_{10}\}$, and we want to distinguish whether $C_1 \cdot C_2 = C_{00} \cdot C_{00}$ or $C_1 \cdot C_2 = C_{01} \cdot C_{10}$.

1. The obfuscator encodes the bits of C_1 and C_2 under the GGH13 map, as well as the bits of all possible inputs $\sigma \in \Sigma$. In particular, we have encodings for σ_1 and σ_2. We homomorphically evaluate U on the encodings of C_1 and σ_1, C_1 and σ_2, C_2 and σ_1 and C_2 and σ_2.[1] These are honest partial evaluations of the circuit $\tilde{U}$ on input σ_1 and σ_2, so we can perform these evaluations (in particular, there will not be incompatibilities of encodings levels). We obtain four pairs of encodings $(R_{b_1b_2} = [r_{b_1b_2}]_{\boldsymbol{v}_{b_1b_2}}, W_{b_1b_2} = [r_{b_1b_2} \cdot w_{b_1b_2}]_{\boldsymbol{v}_{b_1b_2}})$, for $b_1, b_2 \in \{1, 2\}^2$, where

   $$w_{b_1b_2} = (y_{b_1}, C_{b_1}(\sigma_{b_2}), \rho_{b_1b_2}).$$

   Recall that the y part of the encoding does not depend on the input σ, so this is independent of b_2 for our notations.

2. A honest evaluator of the obfuscated program would then multiply the encodings W_{11} and W_{21} (of $C_1(\sigma_1)$ and $C_2(\sigma_1)$) and the encodings W_{12} and W_{22} (of $C_1(\sigma_2)$ and $C_2(\sigma_2)$). However, in order to distinguish which circuit has been obfuscated, we do not perform these honest computations. Instead, following the idea of the mixed input attack described in Section 6.3.3, we compute $W_{11} \cdot W_{22}$ and $W_{12} \cdot W_{21}$ (and we do the same for the r part). We then obtain two encodings $\widetilde{W}_1$ and $\widetilde{W}_2$ of

   $$\begin{aligned} \tilde{w}_1 &:= (y^*, C_1(\sigma_1) \cdot C_2(\sigma_2), \rho_{11}\rho_{22}) \\ \text{and} \quad \tilde{w}_2 &:= (y^*, C_1(\sigma_2) \cdot C_2(\sigma_1), \rho_{12}\rho_{21}) \end{aligned}$$

   at levels $\boldsymbol{v}_{11} + \boldsymbol{v}_{22}$ and $\boldsymbol{v}_{12} + \boldsymbol{v}_{21}$ respectively. Note that the first slot of the encodings contains y^*, as it would for a honest evaluation.

3. We then complete the computation as if $\widetilde{W}_1$ was an honest evaluation on σ_1 and $\widetilde{W}_2$ was an honest evaluation on σ_2. That is, we first multiply $\widetilde{W}_1$ by $\widehat{W_{\sigma_1}}$ and $\widetilde{W}_2$ by $\widehat{W_{\sigma_2}}$ to cancel the third slot. We obtain two encodings $\widehat{W}_1$ and $\widehat{W}_2$ of

   $$\begin{aligned} \widehat{w}_1 &:= (y^* \cdot \widehat{y}, \widehat{\alpha} \cdot C_1(\sigma_1) \cdot C_2(\sigma_2), 0) \\ \text{and} \quad \widehat{w}_2 &:= (y^* \cdot \widehat{y}, \widehat{\alpha} \cdot C_1(\sigma_2) \cdot C_2(\sigma_1), 0) \end{aligned}$$

   at levels $\boldsymbol{v}_{11} + \boldsymbol{v}_{22} + \widehat{\boldsymbol{v}_{\sigma_1}}$ and $\boldsymbol{v}_{12} + \boldsymbol{v}_{21} + \widehat{\boldsymbol{v}_{\sigma_2}}$, respectively.

[1] Recall that $U(\sigma, C) = C(\sigma)$ and the universal circuit we chose is $\tilde{U}(\sigma, C_1 \cdot C_2) = U(\sigma, C_1) \cdot U(\sigma, C_2)$.

4. Finally, we cancel the first slot by subtracting $\overline{W}$ of the encodings $\widehat{W_1}$ and $\widehat{W_2}$ obtained above. Note that this subtraction is between encodings that are not at the same level (for both honest and dishonest evaluations), so the resulting level is the union of the levels of both parts of the subtraction. We obtain two encodings $\overline{W}_1$ and $\overline{W}_2$ of

$$\begin{aligned} \overline{w}_1 &:= (0, \widehat{\alpha} \cdot C_1(\sigma_1) \cdot C_2(\sigma_2), 0) \\ \text{and } \overline{w}_2 &:= (0, \widehat{\alpha} \cdot C_1(\sigma_2) \cdot C_2(\sigma_1), 0) \end{aligned}$$

   at levels $\boldsymbol{v}_{11} + \boldsymbol{v}_{22} + \widehat{\boldsymbol{v}_{\sigma_1}} + \overline{\boldsymbol{v}}$ and $\boldsymbol{v}_{12} + \boldsymbol{v}_{21} + \widehat{\boldsymbol{v}_{\sigma_2}} + \overline{\boldsymbol{v}}$, respectively.

5. Now, we would like to zero-test the encodings $\overline{W}_1$ and $\overline{W}_2$ obtained above, but because we mixed the inputs, the levels of the encodings are unlikely to be $\boldsymbol{v}^*$ and we are not able to zero-test. However, we know that $\boldsymbol{v}_{11} + \boldsymbol{v}_{21} + \widehat{\boldsymbol{v}_{\sigma_1}} + \overline{\boldsymbol{v}} = \boldsymbol{v}^*$, because the encoding obtained by honestly evaluating the obfuscated program on σ_1 has this level. In the same way, we know that $\boldsymbol{v}_{12} + \boldsymbol{v}_{22} + \widehat{\boldsymbol{v}_{\sigma_2}} + \overline{\boldsymbol{v}} = \boldsymbol{v}^*$. Hence, the level of the product $\overline{W}_1 \cdot \overline{W}_2$ is $2\boldsymbol{v}^*$. Using our p'_{zt} parameter, we can then test whether its numerator is a multiple of g^2 or not.

   - In the case where $C_1 \cdot C_2 = C_{00} \cdot C_{00}$, we have $\overline{w}_1 = 0 \bmod g$ and $\overline{w}_2 = 0 \bmod g$. Hence, their product is a multiple of g^2. So the numerator of $\overline{W}_1 \cdot \overline{W}_2$ is a multiple of g^2, and the zero-test using p'_{zt} answers positively.
   - In the case where $C_1 \cdot C_2 = C_{01} \cdot C_{10}$, we have $\overline{w}_1 = 0 \bmod g$ and $\overline{w}_2 \neq 0 \bmod g$. So the product is a multiple of g^2 if and only if $\overline{w}_1$ is a multiple of g^2, which is very unlikely ($\overline{w}_1$ is obtained by subtracting two values that are equal modulo g_1, so this is a multiple of g_1 but this is unlikely to be a multiple of g_1^2).[2] Hence, the numerator of $\overline{W}_1 \cdot \overline{W}_2$ will not be a multiple of g^2 (with high probability), and the zero-test using p'_{zt} will fail.

   We can then distinguish between the obfuscated versions of $C_{00} \cdot C_{00}$ and $C_{01} \cdot C_{10}$ in (classical) polynomial time, using our new zero-testing parameter p'_{zt} obtained in quantum polynomial time.

This completes our quantum attack against the circuit obfuscators of [AB15,Zim15,BD16,DGG+18].

[2] Note that even if $\overline{w}_1$ were a multiple of g^2, then, by taking $p'_{zt} = (z^* \cdot g^{-1})^3 \bmod q$, we could mount the same kind of attack, at level $3\boldsymbol{v}^*$ instead of $2\boldsymbol{v}^*$.