Towards Efficient Arithmetic for Ring-LWE based Homomorphic Encryption

Vincent Zucca

SORBONNE UNIVERSITÉ

THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITÉ

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Vincent Zucca

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Towards Efficient Arithmetic for Ring-LWE based Homomorphic Encryption

Thèse dirigée par Jean-Claude Bajard

soutenue publiquement le 25 Juin 2018

après avis des **rapporteurs** :

M. Fabien Laguillaumie	Professeur, Université de Lyon
M. Frederik Vercauteren	Associate Professor, KU Leuven

devant le **jury** composé de :

M. Jean-Claude Bajard	Professeur, Sorbonne Université
Mme. Anne Canteaut	Directrice de Recherche, INRIA Paris
M. Léo Ducas	Researcher, CWI Amsterdam
M. Philippe Elbaz-Vincent	Professeur, Université Grenoble Alpes
Mme. Caroline Fontaine	Chargée de Recherche, CNRS
M. Fabien Laguillaumie	Professeur, Université de Lyon
Mme. Adeline Roux-Langlois	Chargée de Recherche, CNRS
M. Frederik Vercauteren	Associate Professor, KU Leuven

Remerciements

Voici donc venu le temps des remerciements, partie la plus lue d'une thèse et exercice de style libre mais délicat pour l'auteur qui doit veiller à ne pas décevoir les attentes de ceux qui viendront inévitablement y chercher leur nom. Je me dois également, et avant tout, de te remercier toi, lecteur anonyme pour qui cette thèse a été écrite, d'avoir ouvert ce manuscrit, puisses-tu y trouver ce que tu es venu y chercher.

Comme beaucoup de personnes avant moi, et certainement plusieurs après moi, je tiens à remercier en premier lieu J.-C. pour sa bienveillance à mon égard. Il a su me guider et m'accompagner au cours des trois dernières années. Malgré un contexte parfois compliqué, ta bonne humeur, ton enthousiasme permanent et ton humour, bien que souvent controversé, ont été de précieux atouts durant cette thèse. Celle-ci n'aurait pu aboutir sans la confiance que tu m'as accordée, tes conseils avisés et surtout ton aptitude inestimable à remplacer une pression pesante par des pressions pétillantes ;-). Pour le temps que tu m'as accordé, les opportunités de découvrir le monde que tu m'as offertes, je te remercie.

Si ce manuscrit est entre vos mains aujourd'hui, c'est également grâce à Frederik et Fabien qui ont accepté de rapporter cette thèse. Merci de vous être intéressés à mon travail, d'avoir pris le temps de le lire et d'avoir donné votre aval indispensable. Fabien, merci d'avoir pris le temps de lire cette thèse bien que le chiffrement homomorphe ne soit pas ton sujet de prédilection. Frederik, thank you very much for having paid such a close attention to the reading of this thesis and for all your valuable comments. Je tiens également à remercier Anne, Léo, Philippe, Caroline et Adeline d'avoir accepté de faire partie de mon jury et d'avoir fait le déplacement afin d'assister à ma soutenance.

Si j'ai pu commencer cette thèse, je le dois particulièrement à Philippe, qui a non seulement commencé ma formation à Grenoble en 1 avant J.-C., mais a aussi attiré l'attention de Jean-Claude sur moi. Merci d'avoir veillé, même de loin, au bon déroulement de cette thèse en t'impliquant à chacun des points de contrôle : commencement, comité de suivi et enfin soutenance. Puisque j'en suis aux grenoblois, je souhaite également remercier Jean-Guillaume Dumas qui a su m'initier et me donner goût à la recherche durant le stage de master. Enfin, je voudrais remercier Vanessa Vitse, pour son cours d'introduction au chiffrement homomorphe sans lequel je ne serais certainement pas là aujourd'hui.

La partie la plus intéressante et la plus enrichissante d'une thèse provient incontestablement des rencontres que l'on fait, en cela je dois dire que j'ai eu beaucoup de chance.

En arrivant à Jussieu au début de cette thèse, j'ai eu la chance d'être très bien accueilli par tous les occupants du bureau 338 de l'époque : Ivan, Thibaut, Anastasia, Alexandre, Ulrick et Pacôme. La bonne ambiance qui régnait dans ce bureau, due essentiellement à la grande qualité de ses occupants, m'a apporté un confort indéniable lors du commencement de cette thèse. Je me souviendrai longtemps du dévouement d'Ivan, du chapeau de Thibaut, de la bonne humeur permanente de Nastia, des enragements divers et variés d'Alexandre, de la décontraction d'Ulrick et du flegme de Pacôme. Merci à tous les six pour votre accueil, le Baker Street n'est plus aussi attirant depuis que vous êtes partis.

J'ai bien sûr très vite pu élargir mes horizons avec le bureau 325, dont les occupants principaux Charles et Guillaume, mes référents C++, ont été des rencontres au moins aussi formidables que les précédentes. La décontraction de Guillaume et les débats passionnés et quasi-quotidiens vim/emacs entre Charles et Thibaut, sont à mon sens indissociables de ces trois dernières années. Il y a encore de nombreuses personnes que j'ai apprécié avoir rencontrées et qui doivent au moins être mentionnées : Sénan, Ramon, Clothilde, Daisuke, Cécile, Romain et Jocelyn.

This thesis has also been the opportunity to enriched and enlarged my vision of politic, with the meeting of one of the last Trotskyist on Earth, my favorite Argentinian fellow: Matìas. Thank you for the various talks and moments we have shared together, I have enjoyed them more than you think and I am looking forward the next ones. I hope I have been able to make you discover the charms of Provence and our shitty football. I am proud to count you as a friend now, and I hope someday we will have the opportunity to visit your country together.

Les membres de ma vraie équipe : Alexandre, Thomas, Natasha, Jérôme, Anand, Jérémy et Lucas ont été également autant de rencontres appréciables. Alexandre, merci pour ta disponibilité et ton aide notamment lors de notre arrivée dans l'équipe. Je ne doute pas que ton intérêt pour moi au départ ait été principalement motivé par la clim et les fléchettes dans le bureau ;-), mais nous avons quand même bien rigolé, et pas grâce à ton humour vaseux. Nous avons été pendant un long moment les deux seuls vrais permanents de l'équipe. Thomas, merci d'avoir si brillamment pris la relève d'Alexandre entre tes séjours au Japon et surtout merci d'avoir consacré du temps à la relecture du début de cette thèse, tu es le prochain à soutenir ! Jérôme, merci pour ta disponibilité permanente et ton rire si communicatif. Anand, thank you for always being in such a good mood and for the time you have spent correcting my shitty English.

Plusieurs personnes m'ont aidé d'une façon ou d'une autre durant cette thèse, et en premier lieu notre gestionnaire Irphane que je tiens à remercier particulièrement pour son aptitude à régler tous mes problèmes administratifs avec une efficacité sans égale. Il te faudra maintenant trouver quelqu'un d'autre pour t'aider à préparer les différents workshops :p. Merci à Valérie et Christoph pour m'avoir donné la possibilité d'enseigner durant cette thèse. Je remercie également Antoine pour avoir financé la plus grande partie de ma thèse, Tancrède pour m'avoir

consacré un peu de temps au début de celle-ci pour m'aider à orienter mes recherches, Guénaël pour ses blagues potaches et sa bonne humeur, Elias pour sa gentillesse et sa disponibilité, Karine pour avoir accepté de faire partie de mon comité de suivi malgré sa peur des polynômes cyclotomiques et Damien pour sa disponibilité et sa bonne humeur depuis qu'il nous a rejoint.

J'ai également eu la chance de pouvoir faire différents séjours à l'étranger durant cette thèse. Merci à Thomas de m'avoir donné l'opportunité d'aller travailler avec lui en Australie. Thank you Anwar for welcoming me in Waterloo for three months, and thank you Leonel for your welcome in Lisbon. I would like to thank in particular Paulo, first for being such an efficient co-worker, but most importantly for the good moments we have shared together, first in Paris and then in Lisbon and Sintra, I really appreciated them.

Enfin la personne qui m'a probablement le plus aidé durant cette thèse, et qui a été bien plus qu'un simple co-auteur : Julien. Merci de m'avoir quasiment encadré au début de ces trois années ; si je peux soutenir aujourd'hui c'est clairement grâce à toi. D'ailleurs, si cela ne tenait qu'à moi, ton nom figurerait sur la page de garde. Je te remercie également de m'avoir accueilli chez toi pendant mon séjour à Waterloo. Certes, tout n'aura pas été parfait et nous avons essuyé quelques échecs, notamment la v3 :p, mais les apéros et les séances de grimpe avec Nevin, Claire et Monica resteront toujours de bons souvenirs grâce à toi. Monica good luck for supporting him, I did not expect to see you in my defense when we first met one year ago, but I am glad you have been able to join him in France and to come in Paris for my defense. Bref, merci et bonne chance à tous les deux pour Grenoble !

Si une thèse nous donne l'occasion de rencontrer de nouvelles personnes, on a fatalement moins de temps à consacrer aux anciens amis. Je remercie ceux qui ont su s'accrocher et me pardonner mon manque de disponibilité. Merci aux grenoblois de la première heure : Luc, Gauthier, Benoit, Anaïs, Laurène, Clémentine et Léo ; j'espère que nous arriverons à nous voir plus désormais. Mon séjour à Grenoble ayant duré plus longtemps que prévu :-) j'ai eu la chance de rencontrer d'autres personnes toutes aussi sympathiques : Lolo, Cyril, Loïc et Valentin ; là encore j'espère que nous pourrons nous voir plus souvent. Enfin, les derniers mais pas les moindres, les amis du lycée et d'avant qui sont toujours là après toutes ces années loin d'Aix-en-Provence : Benjamin, Timothé, Patrick, Clément et Raphaël. Merci à vous tous d'être encore là aujourd'hui.

Si j'ai pu faire cette thèse c'est également grâce au soutien indéfectible de ma famille. Merci à mes parents, ma soeur, mon grand-père, mes oncles et tantes, cousins et cousines pour avoir toujours été présents même si je ne l'ai malheureusement pas beaucoup été au cours de ces trois dernières années.

Enfin, je tiens à remercier du fond du coeur Marion, qui a su me soutenir et me supporter même dans les moments les plus difficiles de cette thèse. Celle-ci n'a probablement que peu d'intérêt pour toi mais je peux t'assurer qu'elle n'en aurait eu aucun sans toi.

Contents

Introduction

1 Contexte historique et motivations

Bien que déjà utilisée pendant l'Antiquité, la *cryptologie* – étymologiquement science du secret – n'a justifié son statut de science que récemment grâce aux fondations théoriques posées par Shannon à la fin des années 40 ([Sha49]), et l'avènement de la cryptographie à clé publique dans les années 70. Elle se divise en deux disciplines de nature opposées mais complémentaires : la *cryptographie* et la *cryptanalyse*.

La première a pour but de protéger des communications, pour cela il lui faut garantir quatre propriétés : la *confidentialité*, l'*authenticité*, l'*intégrité* et la *non-répudiation* de celles-ci. La première propriété garantie que seuls les participants à la communication aient accès aux messages. L'authenticité permet d'assurer l'identité des entités qui y participent, tandis que l'intégrité est nécessaire afin de garantir qu'un message n'ait pas été altéré par un tiers non autorisé. Enfin la non-répudiation a pour but d'empêcher que l'une des entités participant à la communication puisse nier l'avoir fait. Afin d'assurer ces propriétés en pratique, les participants à la communication doivent être en possession d'un secret ou *clé*.

La cryptanalyse de manière opposée, a pour but de récupérer des informations à partir d'une communication protégée afin de corrompre l'une, ou plusieurs, des quatre propriétés précédentes et ce sans avoir nécessairement connaissance de la clé.

Historiquement la cryptographie avait surtout pour but d'assurer la confidentialité des messages, les autres propriétés sont essentiellement apparues avec la cryptographie moderne et sont d'autant plus importantes de nos jours avec l'internet. La confidentialité d'un message est assurée en utilisant une clé, afin de le rendre inintelligible à toute personne ne connaissant pas la clé. On parle alors de messages *chiffré*.

En cas de guerre, il est essentiel de s'assurer qu'un message intercepté demeure inintelligible pour l'adversaire. C'est pourquoi historiquement les innovations en matière de chiffrement et de cryptanalyse sont principalement apparues au cours de différents conflits. Parmi les chiffrements célèbres nous pouvons citer la scytale utilisée par les Spartiates lors de la guerre du Péloponnèse, ou encore le chiffrement de César utilisé par les Romains lors de la guerre des Gaules.

Plus récemment, lors des deux grands conflits du XX$^{\text{ème}}$ siècle, la cryptologie eut une importance décisive. En 1917, les renseignements Britanniques ont intercepté et déchiffré

un télégramme du ministre allemand des affaires étrangères, Arthur Zimmermann, envoyé à l'ambassadeur allemand au Mexique dans lequel l'Allemagne proposait une alliance au Mexique. Les termes de cette alliance spécifiaient que le Mexique devrait envahir le sud des États-Unis, si jamais ceux-ci venaient à abandonner leur neutralité, afin de les empêcher d'intervenir en Europe. La révélation de ce télégramme poussa les États-Unis à déclarer la guerre à l'Allemagne et précipita la fin du conflit. Lors de la deuxième guerre mondiale, la machine Enigma, utilisée par les Allemands pour chiffrer leurs communications fut le premier exemple de chiffrement mécanique de l'histoire. Sa cryptanalyse, par l'équipe d'Alan Turing en 1940 nécessita quant à elle la construction du premier ordinateur de l'histoire. Une fois le chiffrement cassé, les Alliés disposèrent d'un avantage certain sur la suite du conflit. La cryptologie venait d'entrer dans l'ère moderne.

Jusqu'à la seconde moitié du XX$^{\text{ème}}$ siècle, la même clé était utilisée pour chiffrer et déchiffrer les messages ; on parle alors de cryptographie *symétrique*. Ce type de chiffrement souffre toutefois d'un inconvénient majeur puisqu'il nécessite que les partis souhaitant communiquer aient échangé la clé de chiffrement, de manière sécurisée, avant le début de la communication. Le problème est alors d'échanger cette clé de manière sécurisée. Ce problème fut solutionné dans les années 70 par l'invention du concept de la cryptographie *asymétrique*, ou encore cryptographie à *clé publique*, par W. Diffie et M. Hellman ([DH76]), et de manière moins connue par R. Merkle ([Mer78]).

La cryptographie à clé publique repose sur le principe que la personne qui doit déchiffrer le message génère deux clés. La première est une *clé secrète*, qui comme son nom l'indique doit rester secrète, la seconde est une clé qui est rendue publique. Cette *clé publique* est utilisée par quiconque souhaitant envoyer un message, de manière sécurisée, au détenteur de la clé secrète afin de chiffrer son message. En revanche, ce dernier est le seul à être capable de déchiffrer le message grâce à la clé secrète. Si ces deux clés sont bien évidemment liées entre elles, on s'assure qu'on ne puisse pas retrouver facilement la clé secrète à partir de la clé publique en faisant en sorte que cela revienne à résoudre un problème calculatoire difficile.

Une bonne analogie à la cryptographie à clé publique est une boite aux lettres. Tout le monde peut facilement glisser un message dans une boite aux lettres. Par contre il est très difficile de récupérer les messages si l'on ne possède pas la clé qui l'ouvre, seul le propriétaire de la boite est en mesure de récupérer les messages facilement.

Dans le contexte de la cryptologie, les mots «facile» ou «difficile» doivent être compris au sens calculatoire du terme, facile signifiant qu'il existe un algorithme permettant de résoudre n'importe quelle instance du problème en temps polynomial tandis que difficile signifie que ce n'est pas facile. Pour des raisons de sécurité il faut que retrouver la clé secrète à partir de la clé publique soit difficile, mais pour des raisons pratiques on souhaite pouvoir déduire la clé publique de la clé secrète facilement.

Ainsi, la cryptographie à clé publique repose sur des problèmes calculatoires qui sont faciles à résoudre dans un sens mais difficile dans l'autre. Un bon exemple d'un tel problème

est la multiplication/factorisation de deux nombres entiers. Multiplier deux entiers se fait facilement, par exemple il n'est pas difficile de trouver que $41 \times 37 = 1517$, en revanche il est moins évident de trouver que 2021 est en fait le produit de 43 et de 47.

C'est précisément en s'appuyant sur ce problème qu'en 1978, R. Rivest, A. Shamir et L. Adleman ont créé leur système de chiffrement, baptisé `RSA`, qui fut la première construction concrète d'un chiffrement à clé publique ([RSA78]). Une propriété notable du `RSA` est que lorsque l'on multiplie deux chiffrés, de deux messages différents, entre eux on obtient alors un chiffré du produit des deux messages. Cette propriété qui permet d'effectuer des opérations sur les messages directement à travers leurs chiffrés, sans avoir à les déchiffrer au préalable, est appelée propriété *homomorphe*.

Dès lors, la question s'est posée de savoir s'il était possible, ou non, de construire un schéma généralisant la propriété homomorphe multiplicative du `RSA`, c'est à dire un chiffrement permettant d'effectuer tout type d'opérations sur les messages directement à travers leurs chiffrés ([RAD78]). On parlera plus tard de *chiffrement complètement homomorphe*.

La question de savoir s'il était possible ou non de construire un schéma de chiffrement complètement homomorphe resta ouverte durant de nombreuses années. Sachant que n'importe quelle fonction continue peut s'approximer par un polynôme sur un compact, il suffit en pratique de pouvoir évaluer des polynômes de manière homomorphique. Pour cela il est seulement nécessaire de disposer d'additions et de multiplications homomorphiques. Plusieurs schémas de chiffrement possédant, comme le `RSA`, une des deux propriétés furent construits dans les années qui suivirent, par exemple ElGamal pour la multiplication ([ElG85]) et Paillier pour l'addition ([Pai99]).

Il a fallu attendre 2005 et les travaux de D. Boneh, E. Goh et K. Nissim pour obtenir la première construction d'un chiffrement possédant les deux propriétés ([BGN05]). Toutefois, cette construction souffre d'un inconvénient majeur puisqu'elle ne permet l'évaluation que d'une seule multiplication de manière homomorphique. Le problème fut finalement solutionné par C. Gentry en 2009, qui proposa la première construction d'un schéma de chiffrement complètement homomorphe permettant l'évaluation d'un nombre arbitraire d'additions et de multiplications et résolvant ainsi un problème de plus de trente ans ([Gen09]).

Cette construction, basée sur les réseaux euclidiens, se déroule en deux temps. La première étape est la construction d'un schéma de base permettant l'évaluation d'un nombre limité d'additions et de multiplications de manière homomorphique, on parle alors de schéma *presque homomorphe*. Comme chaque chiffré de ce genre de schéma contient un bruit, on parle de chiffrement *bruité*. Après chaque opération homomorphe, ce bruit va grossir jusqu'à atteindre une certaine taille au delà de laquelle il ne sera plus possible de déchiffrer le message correctement, d'où le nombre limité d'opérations. La seconde étape est la construction d'une procédure permettant de réduire la taille du bruit d'un chiffré lorsque celle-ci devient trop importante. Cette technique, appelée «*bootstrapping*», consiste à évaluer homomorphiquement le circuit de déchiffrement du schéma. Ainsi, à la fin de la procédure, on obtient un nouveau chiffré du même message, avec un bruit de même taille que si l'on avait évalué homomorphique-

ment le circuit de déchiffrement à partir d'un chiffré «frais». Sachant que l'augmentation du bruit après une multiplication est plus importante qu'après une addition, si le schéma de base permet d'effectuer homomorphiquement une multiplication en plus de l'évaluation de son circuit de déchiffrement, il est dit «bootstrappable» et peut donc être transformé en un schéma complètement homomorphe.

De nos jours, ce genre de chiffrement pourrait avoir un impact considérable sur la protection de la vie privée notamment avec l'essor du «cloud computing». Le chiffrement homomorphe offre une solution concrète à différents problèmes apparaissant en cryptographie comme le calcul multipartite sécurisé, la délégation de calculs ou encore le vote électronique. Malheureusement, bien que théoriquement correcte, la première construction de Gentry requiert de tels paramètres qu'elle est inutilisable en pratique.

C'est pourquoi, dans les années qui suivirent, la communauté consacra des efforts importants à améliorer cette première construction de sorte à la rendre implémentable, ce qui fut accomplit en 2011 ([GH11]). Cette première implémentation fut toutefois rapidement rendue obsolète par l'arrivée des schémas, dit de deuxième génération, qui furent construit les années suivantes. Citons par exemple : Brakerski-Gentry-Vaikuntanathan (BGV) ([BGV12]), Brakerski «scale-invariant» ([Bra12]) ou encore Gentry-Sahai-Waters (GSW) ([GSW13]). Les meilleures performances obtenues par les schémas susmentionnés sont directement liées au problème sous-jacent sur lequel leur sécurité est basée : le Learning With Errors (LWE).

Ce problème, basé sur les réseaux euclidiens et introduit par O. Regev en 2005 ([Reg05]), est actuellement la structure de base sur laquelle une grande partie des schémas de chiffrement homomorphe sont construits. De manière informelle ce problème consiste à la résolution d'un système linéaire «bruité». Plus précisément, considérons une matrice $\boldsymbol{A}$ et un vecteur $\boldsymbol{s}$ à coefficients entiers dans un grand intervalle centré en zéro $[q/2, q/2[$. Connaissant la matrice $\boldsymbol{A}$, la version «décisionnelle» du problème consiste à être capable de déterminer si un vecteur $\boldsymbol{b}$ a ses coefficients tirés uniformément dans $[-q/, q/2[$ ou bien s'il est tel que $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \pmod{q}$, où $\boldsymbol{e}$ est un vecteur de «bruit», i.e. de petits coefficients. De manière équivalente la version «recherche» consiste à retrouver le vecteur $\boldsymbol{s}$ à partir du couple $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \pmod{q})$. Notons l'importance du vecteur $\boldsymbol{e}$, sans lequel ce problème se réduirait une résolution de système linéaire.

Bien que permettant une augmentation significative des performances par rapport à la première construction de Gentry, la structure sous-jacente du LWE est loin d'être satisfaisante en pratique. En effet, pour obtenir des niveaux de sécurité suffisants et un schéma «bootstrappable», il est nécessaire d'utiliser des matrices et des vecteurs de grande taille avec de gros coefficients de sorte que les temps de calculs, et les coûts en mémoire, pour manipuler les éléments deviennent trop importants.

C'est pourquoi, très rapidement, la communauté académique s'est mise à considérer différentes variantes de ce problème, et en particulier sa version sur les anneaux appelée Ring-Learning With Errors (Ring-LWE) ou encore Learning With Errors over Rings qui est basée

sur les réseaux idéaux ([LPR10]). Dans cette version, la matrice A du LWE est remplacée par un élément $\boldsymbol{a}$ de l'anneau $\mathcal{R}_q = (\mathbb{Z}/q\mathbb{Z})[X]/(\boldsymbol{f})$ où $\boldsymbol{f}$ est un polynôme unitaire et irréductible de degré n. Les vecteurs $\boldsymbol{s}$ et $\boldsymbol{e}$ sont à présent considérés comme des éléments de $\mathcal{R}_q$. Le problème est alors similaire : déterminer si un couple d'éléments $(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}_q^2$ a été tiré de manière uniforme dans $\mathcal{R}_q$, ou bien s'il existe des éléments $\boldsymbol{s}$ et $\boldsymbol{e}$ tels que $\boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}$. Il y a deux bénéfices principaux à cette variante :

- $\boldsymbol{a}$ est désormais représenté par n coefficients au lieu de n^2, ce qui permet de gagner un facteur n sur la complexité en mémoire ;
- plutôt que d'avoir à effectuer des produits matrice vecteur (complexité asymptotique $\mathcal{O}(n^2)$), on multiplie à présent des éléments de $\mathcal{R}_q$ ce qui peut être fait plus efficacement (complexité asymptotique $\mathcal{O}(n \log n)$), en particulier lorsque $\boldsymbol{f}(X) = X^n + 1$ pour n une puissance de deux.

Les schémas de chiffrements homomorphes basés sur le LWE se transcrivent assez naturellement au Ring-LWE, ce qui permet d'améliorer significativement leurs performances. Le schéma `BGV` a été développé directement pour les deux problèmes, la construction de Brakerski a été adaptée au Ring-LWE par J. Fan et F. Vercauteren et fut renommée d'après ses auteurs `FV` ([FV12]), l'adaptation de GSW fut, quant à elle, renommée SHIELD ([KGV16]).

Les problèmes du LWE et du Ring-LWE se réduisent tout deux, sous certaines conditions, à des problèmes difficiles sur les réseaux; euclidiens et idéaux respectivement. Pour l'instant aucun algorithme permettant de résoudre le Ring-LWE plus efficacement que le LWE n'est connu, en fait la cryptanalyse concrète d'une instance de Ring-LWE se fait en cryptanalysant une instance de LWE déduite de cette dernière. Dans ce cas les coefficients de $\boldsymbol{a} \in \mathcal{R}_q$ forment la première colonne de la matrice A et les colonnes restantes se déduisent à partir de la première en utilisant la structure du polynôme $\boldsymbol{f}$.

Néanmoins les avantages apportés par le Ring-LWE ne sont pas sans conséquence potentielle. En effet, il est fortement soupçonné que la structure additionnelle présente dans les réseaux idéaux, qui provient du polynôme $\boldsymbol{f}$, puisse être exploitée afin de résoudre des problèmes dans ce genre de réseaux de manière plus efficace. C'est pourquoi, et ce bien qu'on ne sache toujours pas si cette structure peut être exploitée ou pas, il est fortement pressenti que le problème du Ring-LWE est moins difficile que celui du LWE.

En pratique le bootstrapping de Gentry est la procédure la plus coûteuse, ainsi on essaye généralement de l'utiliser le moins possible. Une façon de procéder est de sélectionner des paramètres de schéma presque homomorphe de sorte à pouvoir effectuer toutes les opérations requises par une application particulière, et connue à l'avance, sans utiliser le bootstrapping ([BGV12]).

Toutefois, dans un travail remarquable, L. Ducas et D. Micciancio ont construit un schéma, appelé FHEW, basé sur le Ring-LWE avec lequel le bootstrapping peut être effectué de manière efficace – moins d'une seconde ([DM15]). En introduisant une variante du Ring-LWE sur le tore réel, Chillotti et al. ont amélioré FHEW, qui est devenu TFHE, jusqu'à obtenir un temps

de 0.1 seconde pour le bootstrapping ([CGGI16]). Dans leur contexte, le bootstrapping est effectué après chaque évaluation d'une porte logique, ils peuvent ainsi évaluer homomorphiquement autant de portes qu'ils le souhaitent obtenant ainsi un chiffrement complètement homomorphe très compétitif.

Enfin, il est probablement important de mentionner qu'après des années de recherche et de développement dans les laboratoires académiques, l'ordinateur quantique n'est plus un rêve fantaisiste. Son arrivée prochaine va confronter la cryptologie moderne à la première grande révolution de sa jeune histoire.

En effet, avec un tel ordinateur, les problèmes calculatoires sur lesquels repose la sécurité des systèmes de chiffrements utilisés à l'heure actuelle (factorisation et logarithme discret) pourront être résolus en temps polynomial ([Sho97]). Ainsi, tous les chiffrements basés sur ces problèmes ne seront plus sûrs. Il devient donc nécessaire de développer de nouveaux chiffrements qui soient basés sur des problèmes que l'ordinateur quantique ne pourra pas résoudre en temps polynomial. L'appel récent de la National Institute of Standards and Technology (NIST), afin de poser les bases des futurs standards de la cryptographie post-quantique va dans ce sens.

Dans cette perspective, les problèmes basés sur les réseaux (euclidiens ou idéaux) sont de bons candidats puisqu'on ne connaît pas, à ce jour, d'algorithme quantique permettant de les résoudre de manière significativement plus efficace. Ainsi le chiffrement homomorphe n'est pas compromis à moyen terme par l'arrivée de l'ordinateur quantique. En revanche, l'utilisation de ce genre de chiffrement est toujours limitée en pratique par les coûts importants qu'il nécessite, que ce soit en temps ou en mémoire.

2 But et organisation de la thèse

Le but de cette thèse est d'optimiser l'arithmétique sous-jacente à plusieurs schémas de chiffrements presque homomorphes basés sur le problème du Ring-LWE afin d'améliorer leur performances. Nous espérons contribuer ainsi à lever une partie des barrières qui empêchent toujours l'utilisation de ce type de chiffrement en pratique. Pour cela, nous nous sommes concentrés sur deux points :

- l'optimisation de l'arithmétique utilisée par les primitives de ces schémas, qu'il s'agisse d'opérations intrinsèques à leurs primitives, ou bien d'opérations requises dans le contexte plus général du Ring-LWE ;
- optimiser les calculs spécifiques à une utilisation concrète de ce type de chiffrement, que ce soit au niveau des algorithmes utilisés ou de leurs implémentations.

Le chapitre 1 a pour but d'introduire les différentes notions nécessaires à la bonne compréhension de la suite de ce manuscrit. Pour cela nous commençons par introduire brièvement le problème du Ring-LWE dans le cas des corps cyclotomiques tout en mettant en évidence

son lien avec les réseaux idéaux. Dans une deuxième partie, nous présentons deux des schémas de chiffrements homomorphes les plus couramment utilisés : `BGV` et `FV`, tous deux basés sur le problème du Ring-LWE. Enfin dans une dernière partie, nous présentons les différents outils utilisés afin d'effectuer les opérations requises par ce genre de chiffrement de manière efficace.

La première contribution ([BEHZ17]) de cette thèse est présentée dans le chapitre 2. Ce travail s'est concentré sur l'utilisation du système de représentation des nombres par les restes (RNS). La représentation RNS permet d'effectuer des additions et multiplications sur de grands nombres de manière très efficace, ce qui est très utile dans le contexte du chiffrement homomorphe. Malheureusement, d'autres opérations comme par exemple les comparaisons ne peuvent pas être effectuées directement en RNS. En surmontant les limitations de cette représentation, nous avons réussi à effectuer l'ensemble des calculs nécessaires aux différentes procédures des schémas de chiffrement homomorphe de type «scale-invariant» comme `FV`. Cela s'est traduit par une amélioration remarquable des performances de ces schémas.

L'efficacité de l'arithmétique dans l'espace $\mathcal{R}_q = (\mathbb{Z}/q\mathbb{Z})[X]/(\boldsymbol{f})$ dépend fortement de $\boldsymbol{f}$ notamment à cause de la réduction modulo ce polynôme. Cette efficacité est maximale lorsque $\boldsymbol{f}$ est un cyclotomique de la forme X^n+1. En revanche ce type de polynômes entraîne d'autres limitations qui, selon l'application voulue, peuvent nécessiter de considérer d'autres types de cyclotomiques. Le chapitre 3, présente un travail dans lequel nous nous sommes concentrés à améliorer l'efficacité de l'opération de réduction par le polynôme $\boldsymbol{f}$, lorsque celui-ci est un cyclotomique différent de $X^n + 1$ ([BEH+18]). Cette opération étant nécessaire pour chaque produit d'éléments dans $\mathcal{R}_q$, son optimisation a permis d'augmenter significativement les performances des primitives des schémas `BGV` et `FV` dans ces cas là.

Finalement dans le quatrième et dernier chapitre nous présentons une application concrète d'utilisation de chiffrement homomorphe pour la classification de données privées ([BMSZ18]). De nos jours ce genre de classification utilise de plus en plus des méthodes d'apprentissages automatiques et notamment des machines à vecteurs de support (MVS). Évaluer homomorphiquement une MVS permet de classer des données sans jamais avoir à révéler celles-ci. Nous commençons par présenter le fonctionnement de ces MVS qui sont des techniques appartenant au domaine de l'apprentissage supervisé. Dans un deuxième temps nous exposons les méthodes algorithmiques que nous avons utilisées afin d'effectuer homomorphiquement les opérations requises par ces techniques. L'efficacité de nos méthodes est illustrée par des implémentations sur différents type d'architectures.

Enfin dans une dernière partie nous présentons les conclusions de cette thèse ainsi que les perspectives qu'elle nous permet d'envisager.

Introduction

1 Historical context and motivations

Although already used during ancient times, cryptology – etymologically *science of secret* – only justified its science status recently with its theoritical foundations laid out by Shannon at the end of the 1940s ([Sha49]), and the advent of public-key cryptography (1970s). It can be split into two mains areas of study which are opposite by nature but complementary: *cryptography* and *cryptanalysis*.

The first is about protecting communications from unauthorized third parties, requiring four properties: *confidentiality*, *authentication*, *data integrity* and *non-repudiation*. Confidentiality guaranties that only authorized people can access the messages. The purpose of authentication is to certify the identity of the participants, while data integrity is required to ensure that messages were not corrupted by any unauthorized third party. Finally non-repudiation prevents participants in the communication from falsy denying their implication. In practice, participants of a communication need to possess a secret, also called *key*, to ensure these properties.

On the other hand, cryptanalysis aims at retrieving some information from a secure communication in order to corrupt one, or several, of the previous four properties and this, without necessarily knowing the key.

Historically cryptography was mainly about ensuring the confidentiality of messages. The other properties too are integral to modern cryptography and particularly in the current internet era. Confidentiality of a message is ensured by converting it into an unintelligible text, called *ciphertext*, by using a key. From there, the message can only be understood by people with knowledge of the key.

During war, it is essential to ensure that encrypted messages remain undecipherable to the adversary in the eventuality where they are intercepted. This is why historically most progress in encryption techniques and cryptanalysis appeared during different conflicts. Among the famous classical encryption methods we can mention the scytale used by the Spartans during the Peloponnesian War, and the Caesar cipher used by the Romans during the Gallic Wars.

More recently, during the two major conflicts of the XX$^{\text{th}}$ century, cryptology has been decisive. In 1917, British intelligence intercepted and decrypted a telegram sent by the German foreign secretary, Arthur Zimmermann, to the German ambassador in Mexico in which

Germany was offering an alliance to Mexico. The terms of this alliance specified that Mexico should invade the south of the United States in order to prevent Americans from taking action in Europe in case the United States dropped their neutrality. Once the United States were aware of the content of this telegram they declared war to Germany which sped up the end of the conflict. During World War II the Enigma machine, used by the Germans to encipher their communications, was the first mechanic cipher machine in history. Its cryptanalysis by Alan Turing's team in 1940 has itself required the construction of the first computer in history. Once Enigma was broken, the Allies had a significant advantage for the rest of the conflict. Cryptology just entered its modern era.

Until the second half of the XX$^{\text{th}}$ century, the same key was used to encrypt and decrypt messages: the case referred to as *symmetric* cryptography. However symmetric cryptography suffers from a major drawback since it requires the parties wishing to communicate have exchanged the cipher key before the communication starts. The problem is therefore to exchange this key in a secure way. This was addressed in the seventies when W. Diffie and M. Hellman proposed the concept of *asymmetric* cryptography, also called *public key* cryptography ([DH76]). One should probably mention it was also proposed, roughly at the same time, by R. Merkle even though it is less well-known ([Mer78]).

The principle of public key cryptography is that the person who needs to decrypt messages generates two keys. The first one must remain secret while the second one is made public. The *public key* is used to encrypt messages by anyone willing to communicate with the owner of the *secret key*. However the owner of the secret key is the only one able to decrypt the messages encrypted with the corresponding public key. Of course the two keys are related, but it is made sure that one cannot retrieve easily the secret key from the public key unless by solving a difficult computationnal problem.

An analogy for thinking about public key cryptography is a mail box. Everyone can easily put a letter in a mail box, however it is hard to get the letters which are inside the box if you do not have its key. Only the owner of the mail box can easily take the letters inside it.

In the context of cryptology, "easy" and "difficult" must be understood in terms of computational complexities. "Easy" means that there exists an algorithm which can solve any instantiation of the problem in polynomial time while "difficult" means that it is not easy. If for security reasons it must be difficult to retrieve the secret key from the public key, for efficiency reasons deriving the public key from the secret key should be easy.

As a consequence, public key cryptography relies on mathematical problems which are easy in one way but difficult on the other. A good example of such a problem is the multiplication/factorization of integers. Multiplying two integers is quiet easy, for instance it is not hard to find that $41 \times 37 = 1517$, however it is much harder to find out that 2021 is actually the product of 43 and 47.

It is precisely on this problem that in 1978, R. Rivest, A. Shamir and L. Adleman based their encryption/signature scheme, called `RSA`, which was the first concrete public key con-

struction in history ([RSA78]). A noteworthy property of `RSA` is that when multiplying two ciphertexts, encrypting two different messages, one obtains a ciphertext corresponding to the product of the messages. This property, allowing to process messages directly through their ciphertexts without having to decrypt them beforehand is called *homomorphic* property.

From there the question whether or not it is possible to construct an encryption scheme generalizing the homomorphic property of `RSA` arose. This encryption scheme should allow to perform, not only multiplications, but any kind of computations on messages directly through their ciphertexts ([RAD78]). Later this will be referred as Fully Homomorphic Encryption (FHE).

Whether or not it was possible to construct such a scheme remained opened for many years. Knowing that any continuous function can be approximated over a compact domain by a polynomial, in practice the question can be reduced to being able to evaluate polynomials homomorphically. This only requires to perform both additions and multiplications homomorphically. Several encryption schemes allowing to perform one of the two operations like `RSA` were built in the ensuing years: for instance ElGamal for the multiplication ([ElG85]) and Paillier for the addition ([Pai99]).

The first significant step was made in 2005 by D. Boneh, E. Goh and K. Nissim who constructed the first encryption scheme possessing the two homomorphic properties ([BGN05]). Nevertheless their construction suffers from a major drawback since it only allows to perform one homomorphic multiplication. In a major breakthrough C. Gentry in 2009 resolved the problem by constructing the first fully homomorphic scheme, i.e. which allows to perform an arbitrary number of additions and multiplications homomorphically, solving a thirty years old problem ([Gen09]).

Its construction, based on euclidean lattices, works through two steps. The first step is the construction of a scheme allowing to perform a limited number of additions and multiplications. Such restrictive schemes, referred as Somewhat Homomorphic Encryption (SHE), are called "noisy". In a nutshell, ciphertexts of SHE schemes contains a noise which grows after each homomorphic operation. Unfortunately one is only able to decrypt the message correctly as long as the noise remains small enough, which is the reason why one can only perform a limited number of operations. The second step of Gentry's construction is a procedure to "refresh" ciphertexts, i.e. which allows to reduce the size of the noise inherent to a ciphertext. This procedure, called "*bootstrapping*", roughly consists in running homomorphically the decryption function. At the end of the procedure, we obtain a new ciphertext, encrypting the same message, whose inherent noise has the same size as if one had evaluated the decryption circuit from a "fresh" ciphertext. Noise growth after a multiplication is more important than after an addition. Therefore if the SHE scheme allows to perform one multiplication additionally to the evaluation of its decryption procedure, then the scheme is said "bootstrappable" which means that it can be transformed to a fully homomorphic scheme.

Nowadays, the potential impact of such schemes on privacy is considerable, in particu-

lar with the rise of cloud computing. FHE offers a concrete solution to different problems appearing in cryptography such as secure multiparty/outsourced computations or e-voting. Unfortunately, although theoretically correct, the parameters required by the first construction of Gentry make it unusable in practice.

Hence in the years following Gentry's breakthrough, the cryptographic community has devoted important efforts to improve this first construction in order to make it implementable. This was finally accomplished in 2011 by C. Gentry and S. Halevi ([GH11]). This first implementation quickly became outdated with the so-called second generation schemes which appeared in the next few years. We mention for instance: Brakerski-Gentry-Vaikuntanathan (BGV) ([BGV12]), Brakerski scale invariant construction ([Bra12]) and Gentry-Sahai-Waters (GSW) ([GSW13]). The better performances of the aforementioned schemes are directly related to the underlying problem on which their security is based upon: Learning With Errors (LWE).

The LWE problem was introduced in 2005 by O. Regev in his seminal work ([Reg05]), is currently the main structure used for constructing homomorphic encryption schemes. In an informal way this problem consists in solving a "noisy" linear system. More precisely, let us consider a matrix A and a vector $\boldsymbol{s}$ both with integer coefficients lying in a large centered interval $[-q/2, q/2[$. Knowing the matrix A, the "decisional" version of the problem consists in distinguishing whether a vector $\boldsymbol{b}$ has its coefficients sampled uniformly in $[-q/2, q/2[$ or if it is such that $\boldsymbol{b} = A\boldsymbol{s} + \boldsymbol{e} \pmod q$, for a vector $\boldsymbol{e}$ with "small" coefficients. Equivalently the search version of the problem consists in finding the vector $\boldsymbol{s}$ from the couple $(A, \boldsymbol{b} = A\boldsymbol{s} + \boldsymbol{e} \pmod q)$. Note the importance of the noise/error vector $\boldsymbol{e}$, without which this problem would reduce to the resolution of a linear system.

Despite much better performances when compared to Gentry's first construction, in practice the structure underlying the LWE problem is not completely satisfying. Indeed, in order to obtain levels of security high enough, and to make the scheme bootstrappable, one needs to use large size of matrices and vectors with also large coefficients. As a consequence, the time and memory complexities required to manipulate these elements turn out to be quickly unpractical.

Therefore the cryptographic community has started to consider different variants of this problem and in particular its ring version called Ring-Learning With Errors (Ring-LWE), also called Learning With Errors over Rings, which is based on ideal lattices ([LPR10]). In this variant, the matrix A is substituted for an element $\boldsymbol{a}$ lying in the ring $\mathcal{R}_q = (\mathbb{Z}/q\mathbb{Z})[X]/(\boldsymbol{f})$, where $\boldsymbol{f} \in \mathbb{Z}[X]$ is a monic and irreducible polynomial of degree n. The vectors $\boldsymbol{s}$ and $\boldsymbol{e}$ are now also considered as elements of $\mathcal{R}_q$. The problem is similar : determine whether a couple of elements $(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}_q^2$ was sampled uniformly in $\mathcal{R}_q^2$ or if there exists $\boldsymbol{s}$ and $\boldsymbol{e}$ such that $\boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}$. The benefits brought by this variant are twofold:

- $\boldsymbol{a}$ can now be represented by n coefficients instead of n^2 for a matrix, which reduces the memory complexity by a factor n ;

- instead of performing matrix vector products (asymptotic time complexity $O(n^2)$), one now performs product of elements of $\mathcal{R}_q$ which can be done more efficiently (asymptotic time complexity $O(n \log n)$), especially when $f(X) = X^n + 1$ with n a power-of-two.

Homomorphic encryption schemes based on the LWE problem can be adapted quite naturally to Ring-LWE which enhances significantly their performance. The `BGV` scheme has been directly developed to fit on the two problems, Brakerski' scale-invariant construction was adapted to Ring-LWE by J. Fan and F. Vercauteren and named after them `FV` ([FV12]) and GSW' adaptation was named SHIELD ([KGV16]).

Both LWE and Ring-LWE problems reduce, under some conditions, to hard problems on euclidean and ideal lattices respectively. So far no algorithm solving Ring-LWE more efficiently than LWE is known, and actually concrete cryptanalysis of a Ring-LWE instantiation is currently made by constructing an LWE instantiation from it. In this case the coefficients of the element $a \in \mathcal{R}_q$ form the first column of the matrix A and the other columns are derived from the first one by exploiting the structure of the polynomial f.

Nonetheless, the benefits brought by Ring-LWE are not without potential consequences. Indeed there are important suspicions that the additional structure appearing in ideal lattices, which is brought by the polynomial f, can be exploited to solve problems in this kind of lattices more efficiently. Hence, despite it is not known yet whether this structure can be exploited or not, people tends to believe that Ring-LWE is less hard than LWE.

In practice, Gentry's bootstrapping is the most cumbersome procedure, hence one should try to avoid using it as much as possible. One way of avoiding it is to select parameters for an SHE scheme with a targeted application, so that it allows to perform enough operations to evaluate homomorphically the required functions without bootstrapping ([BGV12]).

In a noteworthy work L. Ducas and D. Micciancio have built a scheme, called FHEW, based on the Ring-LWE problem in which the bootstrapping procedure can be performed very efficiently – less than 1 sec ([DM15]). By considering a variant of Ring-LWE on the real torus, FHEW was improved a few years latter by Chillotti et al. (TFHE) reaching execution time of 0.1 sec ([CGGI16]). In their setting, bootstrapping is performed after the evaluation of each logic gate, so that one can evaluate homomorphically as many gates as he wants achieving therefore FHE with very competitive performance.

It is probably important to mention that after years of research and development in academic laboratories, the quantum computer is no longer a fanciful dream. Its impending arrival will confront modern cryptology to the first revolution of its young history.

Indeed with such computer, the problems on which the security of the current cryptosystems are based upon (factorization, discrete logarithm) can be solved in polynomial time ([Sho97]). As a direct consequence, all the cryptographic schemes whose security is based on these problems will not be safe any longer. Therefore it becomes urgent to develop new constructions based on problems which cannot be solved more efficiently with a quantum

computer. The recent call of the national Institute of Standards and Technology (NIST) to lay down the foundations of the future standards of post-quantum cryptography goes in this direction.

In view of this, problems based on lattices (euclidean or ideal) are good candidates since we do not know so far any quantum algorithm to solve them more efficiently than with classical algorithms. This is why the medium-term security of homomorphic encryption schemes is not compromised by the quantum computer. However the important costs in time and memory required in practice by homomorphic schemes are still the major bottleneck to their deployment for real-life applications.

2 Contributions and organization of the report

The purpose of this thesis is to optimize the underlying arithmetic of several SHE schemes based on the Ring-LWE problem in order to enhance their performance. In this way, we hope to break some barriers which still prevent the widespread use of homomorphic encryption. To this end, we have focused on two main approaches:

- optimize the arithmetic used by the primitive of SHE schemes by considering, either the specific operations of these primitives, or the operations needed in the more general framework of Ring-LWE;
- optimize the operations required by a concrete and specific application of homomorphic encryption, both at the algorithmic level and their practical implementations.

Chapter 1 aims at introducing the different notions needed for a correct understanding of the rest of the report. We start by briefly introducing the Ring-LWE problem in the context of cyclotomic fields while highlighting its link with ideal lattices. In a second part we present two of the most currently used homomorphic encryption schemes: `BGV` and `FV` which are both based on the Ring-LWE problem. Finally, we present the different tools used in practice to perform the operations needed by these schemes efficiently.

The first contribution of this thesis ([BEHZ17]), presented in Chapter 2, is a suite of algortihms implementing all the primitives of scale-invariant SHE schemes efficiently within the Residue Number System (RNS) representation. RNS representation permits efficient arithmetic (additions and multiplications) for big numbers which is critical to homomorphic encryption. Unfortunately certain operations, like comparison for instance, cannot be performed directly in RNS. By overcoming these limitations, we have been able to perform all the computations required by the different procedures of scale-invariant schemes like `FV` in RNS representation. This has resulted in a noticeable improvement of the performance of these schemes.

Efficiency of the arithmetic in $\mathcal{R}_q = (\mathbb{Z}/q\mathbb{Z})[X]/(\boldsymbol{f})$, which is the ambient space in Ring-LWE, heavily depends on $\boldsymbol{f}$, in particular because of the reduction modulo this polynomial. This

efficiency is maximal when $\boldsymbol{f}$ is a cyclotomic polynomial of the form $X^n + 1$, but this kind of polynomials suffers from others limitations which, depending on the targeted application, may require to consider other kind of cyclotomics. Chapter 3 presents a work in which we have focused on improving the reduction modulo $\boldsymbol{f}$ procedure, when it is a cyclotomic different from X^n+1 ([BEH+18]). Since this operation is required for each product of elements in $\mathcal{R}_q$, its optimization has led to a significant improvement of the performance of `BGV` and `FV` in these cases.

In the fourth and last chapter we present a concrete use case of homomorphic encryption to the classification of private data ([BMSZ18]). This classification uses advanced machine learning techniques and in particular Support Vector Machine (SVM). Evaluating a SVM homomorphically allows to classify data without revealing them. We start by presenting how SVMs, which are techniques from the supervised learning domain, work. We then describe the algorithmic methods we have devised in order to perform the operations required by SVMs homomorphically. The efficiency of our methods is illustrated by several implementations on different kind of architectures.

Finally in the last chapter, we summarize the results of this thesis together with directions for possible future works.

Chapter 1

Preliminaries

Contents

This chapter aims to familiarize the reader with the general context of this thesis. Therefore we start by introducing the celebrated Ring-LWE problem and its relation to ideal lattices. Afterwards we present two of the most used homomorphic encryption schemes in practice `BGV` and `FV`, both based on Ring-LWE. Finally we give an overview of the different arithmetic tools used to perform computations in these schemes.

1.1 Ring-Learning with Errors problem

In this section we introduce the Ring-LWE problem in the context of cyclotomic fields. Although this problem can be defined and proven hard in any number field [PRSD17], we restrict our exposition to the case of cyclotomic fields, which are among the most used in practice.

1.1.1 Cyclotomic polynomials and cyclotomic number fields

Let $m \geq 1$ be an integer and let us consider $\mathcal{K}_m$ the splitting field of $X^m - 1$ over $\mathbb{Q}$.

Definition 1.1.1. An element $\zeta \in \mathcal{K}_m$ such that $\zeta^m = 1$ is called a *m-th root of unity*. Moreover if ζ is of order m then it is called a *primitive* m-th root of unity.

Remark 1.1.2. In $\mathbb{C}$, the m distinct m-th roots of unity form a multiplicative group of order m hence one can think of $\mathcal{K}_m$ as the smallest extension of $\mathbb{Q}$ containing these elements. Since a finite subgroup of the multiplicative group of a field is cyclic, there exists an element of order m in $\mathcal{K}_m$.

From now, we denote by $\zeta_m \in \mathcal{K}_m$ a primitive m-th root of unity. Since ζ_m generates all the m-th roots of unity, it generates in particular all the other primitive m-th roots of unity thus $\mathcal{K}_m = \mathbb{Q}(\zeta_m)$.

Lemma 1.1.3 ([DF04] section 13.4)**.** Let $\zeta_m \in \mathcal{K}_m$ be a primitive m-th root of unity. For any $0 \leq k < m$, ζ_m^k is a primitive m-th root of unity if and only if k is coprime to m. In particular there are $\varphi(m)$ primitive m-th roots of unity where φ denotes the Euler's totient function.

From the primitive m-th roots of unity we can define the m-th cyclotomic polynomial $\mathbf{\Phi}_m \in \mathcal{K}_m[X]$ as the product of the $(X - \zeta_m)$ where ζ_m ranges over all the primitive m-th roots of unity.

Definition 1.1.4. Let $m \geq 1$ be an integer, the polynomial defined by:

$$\mathbf{\Phi}_m(X) = \prod_{\substack{1 \leq k < m \\ k \wedge m = 1}} (X - \zeta_m^k)$$

is called the m-th *cyclotomic polynomial* and its degree is $n = \varphi(m)$.

We can deduce directly from the definition that $\mathbf{\Phi}_m$ is monic and divides $X^m - 1$; it follows directly that:

Proposition 1.1.5 ([DF04] section 13.6)**.** For any integer $m \geq 1$ we have:

$$X^m - 1 = \prod_{d|m} \mathbf{\Phi}_d(X) \tag{1.1.1}$$

Properties to recursively compute cyclotomic polynomials can be derived from (1.1.1). We do not aim to present those properties but the interested reader can find them in references such as [Lan05] and [DF04]. However the following is essential for our purpose.

Lemma 1.1.6 ([DF04] section 13.6)**.** For any integer $m \geq 1$ the m-th cyclotomic polynomial $\mathbf{\Phi}_m$ is monic and has integer coefficients.

Theorem 1.1.7 ([DF04] section 13.6)**.** For any integer $m \geq 1$ the m-th cyclotomic polynomial $\mathbf{\Phi}_m$ is irreducible over the field of rationals $\mathbb{Q}$.

Corollary 1.1.8 ([DF04] section 13.6)**.** $\mathbf{\Phi}_m$ is the minimal polynomial of any primitive m-th root of unity, hence $\mathcal{K}_m = \mathbb{Q}(\zeta_m) \cong \mathbb{Q}[X]/(\mathbf{\Phi}_m(X))$. The degree of the extension is therefore $[\mathbb{Q}(\zeta_m) : \mathbb{Q}] = \varphi(m)$.

Definition 1.1.9. For an integer $m \geq 1$, the m-th *cyclotomic number field* $\mathcal{K}_m = \mathbb{Q}(\zeta_m)$ is the field extension obtained by adjoining an element ζ_m of multiplicative order m to $\mathbb{Q}$.

Cyclotomic number fields have been intensively studied but for our purpose we are only interested in a few of their properties. The interested reader can nonetheless refer to the more extensive reference [Was97]. We start by introducing the notion of ring of integers.

Definition 1.1.10. A number $\alpha \in \mathbb{C}$ is called an *algebraic integer* if it is a root of a monic polynomial with integer coefficients.

Theorem 1.1.11 ([DF04] section 15.3)**.** The set of algebraic integers of $\mathcal{K}_m$ is a ring and a free $\mathbb{Z}$-module of rank $n = \varphi(m)$.

Proposition 1.1.12 ([Was97] Proposition 1.2)**.** The ring of algebraic integers of $\mathcal{K}_m$ is $\mathbb{Z}[\zeta_m] := \left\{ \sum_{i=0}^{k} a_i \zeta_m^i \ / \ k \in \mathbb{N},\ (a_i)_{i=0}^{k} \in \mathbb{Z} \right\}$.

Remark 1.1.13. The set $\{1, \zeta_m, \ldots, \zeta_m^{n-1}\}$ forms a basis of $\mathbb{Z}[\zeta_m]$ as a free-$\mathbb{Z}$ module called its *power basis*. It corresponds to $\{1, X, \ldots, X^{n-1}\}$ when $\mathbb{Z}[\zeta_m]$ is identified to the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(\mathbf{\Phi}_m(X))$ through the ring isomorphism $\zeta_m \mapsto X$.

We would like to draw the reader's attention on the following important facts.

Lemma 1.1.14 ([LM06] Lemma 3.2)**.** Let $\boldsymbol{f} \in \mathbb{Z}[X]$ a monic irreducible polynomial of degree n then for every non-zero element $\boldsymbol{a} \in \mathbb{Z}[X]/(\boldsymbol{f}(X))$, the elements $\boldsymbol{a}, X \cdot \boldsymbol{a}, \ldots, X^{n-1} \cdot \boldsymbol{a}$ are linearly independent.

As a direct consequence, by identifying $Z[\zeta_m]$ to the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(\mathbf{\Phi}_m(X))$, we can see that every non-zero element $\boldsymbol{a}$ in an ideal of $\mathbb{Z}[\zeta_m]$, generates a shifted basis $\{\boldsymbol{a}, \boldsymbol{a} \cdot \zeta_m, \ldots, \boldsymbol{a} \cdot \zeta_m^{n-1}\}$ of this ideal seen as $\mathbb{Z}$-module.

Corollary 1.1.15. Any non-zero ideal $\mathcal{I} \subseteq \mathbb{Z}[\zeta_m]$ is a free $\mathbb{Z}$-module of rank $n = \varphi(m)$.

Definition 1.1.16. A *fractional ideal* $\mathcal{F}$ is a set of $\mathcal{K}_m$ such that there exists a non-zero $d \in \mathbb{Z}[\zeta_m]$ and $d \cdot \mathcal{F} = \{df \ / \ \forall f \in \mathcal{F}\} \subseteq \mathbb{Z}[\zeta_m]$ forms an ideal of $\mathbb{Z}[\zeta_m]$. Equivalently it can be seen as the subset of $\mathcal{K}_m$ defined by $\mathcal{F} = d^{-1} \cdot \mathcal{I}$ for an ideal $\mathcal{I} \subseteq \mathbb{Z}[\zeta_m]$.

Remark 1.1.17. Note that every ideal of $\mathbb{Z}[\zeta_m]$ is a fractional ideal. In order to differentiate fractional ideals with ideals of $\mathbb{Z}[\zeta_m]$ the latter are called *integral ideals*.

Remark 1.1.18. As fractional ideals are defined through an integral ideal therefore they also form a free $\mathbb{Z}$-module of rank n, their bases being the bases of the corresponding integral ideal multiplied by d^{-1}.

Another noteworthy fact about cyclotomic fields concerns their Galois group.

Definition 1.1.19. Let L be an extension of a field K, an automorphism σ of L over K is a field automorphism of L such that σ coincides with the identity on K.

It is straightforward to show the set of all the field automorphisms of L over K is a group for the function composition operator. When L is a Galois extension of K, this group is called *the Galois group of L over K* and is denoted $\mathrm{Gal}(L/K)$. Since $\mathcal{K}_m$ is the splitting field of $X^m - 1$ over $\mathbb{Q}$ it is Galois and we know exactly the structure of its Galois group.

Theorem 1.1.20 ([Was97] Theorem 2.5)**.** There is a group isomorphism between the Galois group $\mathcal{G} = \mathrm{Gal}(\mathcal{K}_m/\mathbb{Q})$ and $(\mathbb{Z}/m\mathbb{Z})^\times$ which is given by $i \mapsto (\zeta_m \mapsto \zeta_m^i)$. In particular $\mathcal{K}_m$ has abelian Galois group.

1.1.2 Lattices and ideal lattices

Definition 1.1.21. Let $n \geq 1$ be an integer, a *lattice* $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^n$ endowed with a norm $\|\cdot\|$.

Discrete means that for a given norm there exists an n-dimensional ball centered at zero that does not contain any vector of the lattice $\mathcal{L}$ except 0. Therefore there exists a non-zero vector $u \in \mathcal{L}$ which minimizes this norm. This vector is not unique ($-u$ also satisfies this condition) and is called a *shortest vector* of $\mathcal{L}$. One may notice that if a vector $v \in \mathcal{L}$ is linearly dependent with u, i.e. such that $v = \alpha \cdot u$, then $\alpha \in \mathbb{Z}$. Indeed otherwise the non-zero vector $v - \lfloor \alpha \rceil \cdot u \in \mathcal{L}$, with $\lfloor \cdot \rceil$ the rounding to the nearest integer, would have a smaller norm than u which would contradict the definition of u. From there we can build by induction on the degree of liberty of $\mathcal{L}$ a family of linearly independent vectors that generate $\mathcal{L}$.

Definition 1.1.22. A *basis* $\mathcal{B}$ of a lattice $\mathcal{L}$ is a set of $d \leq n$ linearly independent vectors $\{b_1, \ldots, b_d\}$ such that any element of $\mathcal{L}$ can be written as a linear combination, with integer coefficients, of elements of $\mathcal{B}$. The lattice generated by a basis $\mathcal{B}$ is denoted $\mathcal{L}(\mathcal{B})$.

As a direct consequence a lattice of rank d can be seen as a free $\mathbb{Z}$-module of rank d. We can represent it by a matrix $B \in \mathcal{M}_{n,d}(\mathbb{R})$ formed by the column vectors of the basis. One may notice that any non-trivial unimodular transformation (matrix with integer coefficients whose determinant is ± 1) transforms a basis into another. Hence a lattice of rank $d \geq 2$ has an infinite number of bases, all of same cardinality. The common cardinality of the bases is called the *rank* of the lattice while a *full rank* lattice refers to a lattice of rank n. From now on $n = \varphi(m)$ will denote the dimension of $\mathcal{K}_m$ as $\mathbb{Q}$-vector space.

Definition 1.1.23. A *geometric embedding* $\iota : \mathcal{K}_m \hookrightarrow \mathbb{R}^n$ is an injective morphism of additive groups.

Through a geometric embedding ι any fractional ideal can be mapped to $\mathbb{R}^n$ and can therefore be identified to a full rank lattice (remark 1.1.18). However those lattices inherit some additional structure from the absorption property of their generating integral ideal, in order to differentiate them from other lattices, they are called *ideal lattices*.

Definition 1.1.24. Let ι be a geometric embedding, then any fractional ideal $\mathcal{I} \subseteq \mathcal{K}_m$ can be identified through ι to a full rank lattice $\mathcal{L}(\mathcal{I}) \subseteq \mathbb{R}^n$ and this lattice is called an *ideal lattice*.

Remark 1.1.25. Depending on the geometric embedding the lattice will not be the same. Therefore, one should always precise through which one the ideal lattice is considered.

The most intuitive geometric embedding is probably the *coefficient embedding* which represents an element $\boldsymbol{a} = \sum a_i \zeta_m^i \in \mathbb{Z}[\zeta_m]$ through its coordinates in the power basis $(a_0, \ldots, a_{n-1}) \in \mathbb{Z}^n$. Unfortunately, despite its relative simplicity, the coefficient embedding is not adapted from an algebraic point of view since it does not preserve the multiplication and thus the structure of ideals. The *canonical embedding* from algebraic number theory on the other hand, allows to perform multiplication coefficient-wise and thus preserves this structure. Although two embeddings are always related by a linear transformation of $\mathbb{R}^n$, the multiplicative property of the canonical embedding makes it more suited for the study of ideal lattices and enables the security reduction of the Ring-LWE problem ([LPR10]) and thus its theoretical foundation .

Each element $\sigma_i \in \mathcal{G} = \mathrm{Gal}(\mathcal{K}_m/\mathbb{Q})$ defines an embedding from $\mathcal{K}_m$ to $\mathbb{C}$ given by $\zeta_m \mapsto \zeta_m^i$ for $i \in (\mathbb{Z}/m\mathbb{Z})^\times$. Since any polynomial with real coefficients which vanishes on a complex number x also vanishes on its complex conjugate $\overline{x}$, complex embeddings in number fields always come in pairs. Let s_1 be the number of real embeddings and s_2 be the number of pairs of complex embeddings, we have $s_1 + 2s_2 = n$. We define $H \subseteq \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2}$ hereafter:

$$H = \{(x_1, \ldots, x_n) \in \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2} : x_{s_1+s_2+j} = \overline{x_{s_1+j}}, \text{ for } 1 \leq j \leq s_2\} \subseteq \mathbb{C}^n \tag{1.1.2}$$

Definition 1.1.26. Let the elements of $\mathrm{Gal}(\mathcal{K}_m/\mathbb{Q})$ be denoted σ_i with $1 \leq i \leq n$ and ordered such that $\sigma_{s_1+s_2+j} = \overline{\sigma_{s_1+j}}$ with $1 \leq j \leq s_2$, the *canonical embedding* is the map $\sigma : \mathcal{K}_m \hookrightarrow H$ such that $\sigma(\boldsymbol{a}) = (\sigma_1(\boldsymbol{a}), \ldots, \sigma_n(\boldsymbol{a}))$ for any $\boldsymbol{a} \in \mathcal{K}_m$.

Remark 1.1.27. Since the elements σ_i are field homomorphisms, additions and multiplications can be performed coefficient-wise in the canonical embedding.

Remark 1.1.28. General number fields may have some real embedding corresponding to the real roots of their minimal polynomial but in our case $\mathbf{\Phi}_m$ has no real root as soon as $m > 2$. Since the cases $m = 1, 2$ present few interest from a cryptographic point of view, we only consider the case where $m > 2$, thus $\mathcal{K}_m$ has only complex embeddings and $s_1 = 0$.

Remark 1.1.29. For $p \geq 1$, H can be endowed with the ℓ_p norm defined by $\|\boldsymbol{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$ when $p < \infty$ and $\|\boldsymbol{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$ when $p = \infty$. By identifying elements of $\mathcal{K}_m$ with their canonical embedding in H, we can speak of geometric norm on $\mathcal{K}_m$ $\left(\|\boldsymbol{x}\|_p = \|\sigma(\boldsymbol{x})\|_p\right)$.

The attentive reader may have noticed that our definition of the canonical embedding does not match the definition of a geometric embedding as stated in definition 1.1.23 since it embeds into $\mathbb{C}^n$. Elements of H have half of their coordinates fixed by complex conjugation, therefore H has dimension $n/2$ as $\mathbb{C}$-vector space and thus dimension n as $\mathbb{R}$-vector space. In order to work with real lattices, we need to differentiate the real and imaginary parts of the canonical embedding to pull it back to $\mathbb{R}^n$. In the case of cyclotomic fields with $m > 2$, it can be done through the linear transformation whose matrix A is given below:

$$A = \frac{1}{\sqrt{2}} \begin{pmatrix} I_{n/2} & I_{n/2} \\ iI_{n/2} & -iI_{n/2} \end{pmatrix}$$

with I_n the identity matrix of size n, $i = \sqrt{-1}$. The normalization factor $1/\sqrt{2}$ is added in order to preserve the ℓ_2 norm through the transformation A (i.e. such that $\ell_2(Ax) = \ell_2(x)$ for any $x \in H$).

Figures 1 and 2 illustrate on a small example how different the lattices generated by the coefficient and canonical embedding are. One can see for instance that one is sparser than the other and that their number of short vectors is different. The interested reader can find a more detailed study of the relations between these two embeddings in [Bat15].

Example 1.1.30. Let us consider the lattice generated by the ideal $(1 - \zeta_3) \subset \mathbb{Z}[\zeta_3]$ through the coefficient and canonical embeddings.

Through the coefficient embedding $1 - \zeta_3$ corresponds to the vector $\boldsymbol{b}_1$ whose coordinates are $(1, -1) \in \mathcal{R}^2$ and $\zeta_3(1-\zeta_3) = \zeta_3 - (-\zeta_3 - 1) = 2\zeta_3 + 1$ (since $\mathbf{\Phi}_3(X) = X^2 + X + 1$) corresponds to the vector $\boldsymbol{b}_2 = (1, 2)$. These two vectors form a basis of the lattice generated by $(1 - \zeta_3)$ (Lemma 1.1.14) through the coefficient embedding.

However $\sigma(1 - \zeta_3) = (1 - \zeta_3, 1 - \overline{\zeta_3}) = 1/2(3 - i\sqrt{3}, 3 + i\sqrt{3}) \in H$ ($\zeta_3 = e^{2i\pi/3} = -1/2 + i\sqrt{3}/2$) and $\boldsymbol{b}'_1 = \sigma(1-\zeta_3)A^t = 1/\sqrt{2}(3, \sqrt{3})$. Similarly $\sigma(2\zeta_3+1) = (i\sqrt{3}, -i\sqrt{3})$ and $\boldsymbol{b}'_2 = \sigma(2\zeta_3+1)A^t = (0, \sqrt{6})$. Thus $(\boldsymbol{b}'_1, \boldsymbol{b}'_2)$ form a basis of the lattice generated by $1 - \zeta_3$ through the canonical embedding.

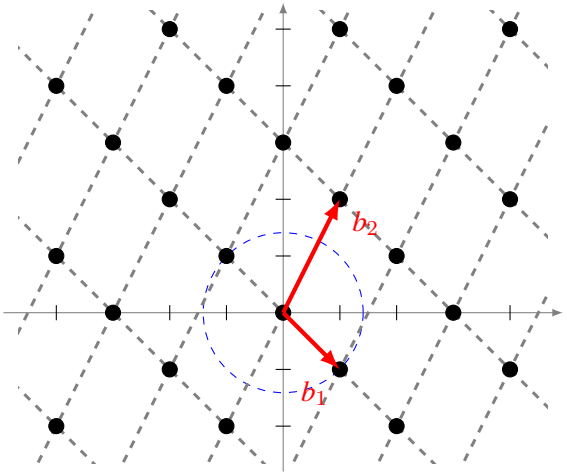


Figure 1: Lattice generated by $(1-\zeta_3) \subset \mathbb{Z}[\zeta_3]$ through the coefficient embedding.

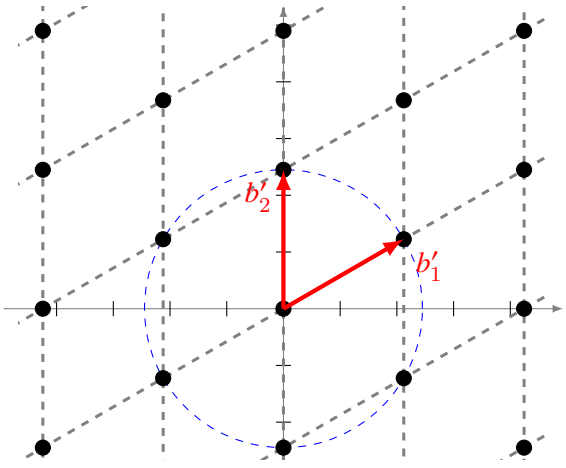


Figure 2: Lattice generated by $(1-\zeta_3) \subset \mathbb{Z}[\zeta_3]$ through the canonical embedding.

1.1.3 Theoretical hardness of Ring-Learning With Errors

Defining the Ring-LWE problem in a formal way would involve notions and considerations on the shape of the error distribution which are beyond the scope of this thesis. For the sake of simplicity, we introduce the Ring-LWE problem and highlight its hardness in an informal way. For further details on the rigorous definition of Ring-LWE the reader should refer to [LPR10]. We start by introducing the Shortest (Independent) Vector Problem (S(I)VP) in the context of ideal lattices. Unless stated otherwise, from now we only consider ideal lattices through the canonical embedding.

For a given norm, the length of a shortest vector in a lattice $\mathcal{L}$ is called the *minimum distance of the lattice* and denoted $\lambda_1(\mathcal{L})$. This notion can be extended to define the sequence of *successive minima*. The k-th minimum of the lattice $\mathcal{L}$ is defined as the smallest positive real number $\lambda_k(\mathcal{L})$ such that there exists k linearly independent vectors of $\mathcal{L}$, with each vector of norm at most $\lambda_k(\mathcal{L})$.

Definition 1.1.31 (SVP and SIVP)**.** Let $\mathcal{K}_m$ be the m-th cyclotomic number field endowed with a norm $\|\cdot\|$ and let $\gamma \geq 1$. The $\mathcal{K}_m$-SVP$_\gamma$ problem in the given norm is: given a fractional ideal $\mathcal{I}$ in $\mathcal{K}_m$ find some non-zero $\boldsymbol{x} \in \mathcal{I}$ such that $\|\boldsymbol{x}\| \leq \gamma \cdot \lambda_1(\mathcal{I})$. The $\mathcal{K}_m$-SIVP$_\gamma$ problem is defined similarly, where the goal is to find n linearly independent elements in $\mathcal{I}$ whose norms are all at most $\gamma \cdot \lambda_n(\mathcal{I})$.

Remark 1.1.32. SVP was proven to be an NP-Hard problem for generic lattices ([vEB81], [Ajt98], [Mic01]), while S(I)VP$_\gamma$ is hard within polynomial factor $\gamma = n^{O(1)}$ ([Ajt96]). Despite considerable efforts it is not known so far whether or not these problems are easier for ideal lattices and the best current strategy to solve them is to use lattice reduction algorithms.

Remark 1.1.33. For convenience we often identify a fractional ideal $\mathcal{I}$ with its embedded lattice $\mathcal{L}(\mathcal{I})$, and speak of the minimum distance of an ideal $\lambda_1(\mathcal{I})$.

Let $m \geq 1$ be an integer and $\mathcal{R} = \mathbb{Z}[X]/(\mathbf{\Phi}_m(X))$ ($\cong \mathbb{Z}[\zeta_m]$) be the ring of integer polynomials modulo $\mathbf{\Phi}_m(X)$. Furthermore for an integer $q \geq 2$, consider the ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} \cong (\mathbb{Z}/q\mathbb{Z})[X]/(\mathbf{\Phi}_m(X))$, each element of $\mathcal{R}_q$ can be represented as a polynomial of degree smaller than n and coefficients in $\{0, \ldots, q-1\}$. Let χ_{err} be an error distribution on $\mathcal{R}$ and $\mathcal{U}_q$ be the uniform distribution on $\mathcal{R}_q$. Given a probability distribution $\mathcal{D}$, we use $x \xleftarrow{\$} \mathcal{D}$ to denote that x is sampled randomly from $\mathcal{D}$.

Definition 1.1.34 (Ring-LWE Distribution)**.** For $\boldsymbol{s} \in \mathcal{R}_q$ and an error distribution χ_{err} over $\mathcal{R}$, a sample from the Ring-LWE distribution $A^q_{s,\chi_{err}}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is the distribution obtained by sampling $\boldsymbol{a} \xleftarrow{\$} \mathcal{U}_q$ and $\boldsymbol{e} \xleftarrow{\$} \chi_{err}$ and outputting $\left(\boldsymbol{a}, [\boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}]_q\right) \in \mathcal{R}_q^2$.

Definition 1.1.35 (Ring-LWE, Decisional)**.** Let $q \geq 2$ be an integer, $\boldsymbol{s} \in \mathcal{R}_q$ and χ_{err} be a distribution over $\mathcal{R}$. The decisional version of the Ring-LWE problem is defined as follows: given access to a polynomial number of independent samples in $\mathcal{R}_q \times \mathcal{R}_q$, determine whether they were drawn from $A^q_{\boldsymbol{s},\chi_{err}}$ or from the uniform distribution on $\mathcal{R}_q \times \mathcal{R}_q$.

Definition 1.1.36 (Ring-LWE, Search)**.** Let $q \geq 2$ be an integer, $\boldsymbol{s} \in \mathcal{R}_q$ and χ_{err} be a distribution over $\mathcal{R}$. The search version of the Ring-LWE problem is defined as follows: given access to a polynomial number of independent samples from the Ring-LWE distribution $A^q_{\boldsymbol{s},\chi_{err}}$, find $\boldsymbol{s}$.

The error term $\boldsymbol{e}$ is supposed to be sampled according to a discrete n-dimensional centered spherical Gaussian distribution in the embedding space H, with each coordinate sampled independently ([LPR10], [CIV16]). A noteworthy case is when m is a power of two, in which case the geometry of the corresponding cyclotomic field allows to sample the error term $\boldsymbol{e}$, as a scaled centered spherical Gaussian, directly in $\mathcal{R}$. We can emphasize two main approaches to transfer the error in $\mathcal{R}$ for a general m. In [LPR13] the authors explain how to invert the canonical embedding map in quasi-linear time allowing therefore to transfer the error term from H to $\mathcal{R}$. The authors of [DD12] proposed an alternative method to generate the error and proved it does not affect notably the difficulty of the problems. Their approach consists in sampling the error according to a centered spherical Gaussian in $\mathbb{Q}[X]/(\boldsymbol{\theta}_m(X))$, where $\boldsymbol{\theta}_m(X) = X^{m/2} + 1$ if m is even and $\boldsymbol{\theta}_m(X) = X^m - 1$ otherwise. From there the error is reduced to $\mathbb{Q}[X]/(\mathbf{\Phi}_m(X))$ with a polynomial reduction and then its coordinates are rounded coefficient-wise to get it in $\mathcal{R}$.

Lyubashevski et al. have shown that the Ring-LWE Search problem reduces to the Ring-LWE Decisional problem [LPR10]. It was later proven that both problems are equivalent [Lyu11] and both reduce to the Shortest (Independent) Vector(s) Problem:

Theorem 1.1.37 ([LPR10])**.** Let m be an integer and q a prime congruent to 1 modulo m. Also let $\alpha = \alpha(\varphi(m)) \in (0,1)$ be a real number such that $\alpha \cdot q > \omega(\sqrt{\log m})$. If there is an algorithm that can solve the decisional Ring-LWE problem for a secret $\boldsymbol{s}$ sampled uniformly in $\mathcal{R}_q = (\mathbb{Z}/q\mathbb{Z})[X]/(\mathbf{\Phi}_m(X))$ and an error distribution $\chi_{err}(\alpha)$; then there exists a quantum

algorithm that runs in time $O(q \cdot \text{poly}(m))$ which solves the S(I)VP$_\gamma$ to within a factor $\gamma \in \tilde{O}(\sqrt{m}/\alpha)$ in any ideal of the ring $\mathbb{Z}[\zeta_m]$.

Remark 1.1.38. The conditions on q to be a prime and congruent to 1 modulo m in the theorem have been removed in [LS12] so that q has only to be an integer greater than 2.

Remark 1.1.39. The secret s should formally be considered as an element of $\mathcal{R}^\vee$ the dual fractional ideal of $\mathcal{R}$, however the authors of [DD12] have also shown that it could be chosen directly uniform in $\mathcal{R}_q$ without making the problem any easier.

Selecting parameters for a secure Ring-LWE instantiation requires to take many parameters in consideration, in particular the width of the error distribution, and should be done carefully. In [Pei16], Peikert exposes several instantiations of Ring-LWE which are insecure because the error distribution is not well chosen.

1.1.4 Practical considerations

Conditions of the Ring-LWE reduction to S(I)VP ensure an asymptotic security however in practice we need to have an idea of the concrete hardness of an instantiation to estimate its security. Concrete cryptanalysis of lattice based cryptography is beyond the scope of this thesis. However having a basic understanding of the relation between the parameters and the security of a Ring-LWE instantiation is essential to avoid gross errors in parameters selection. We remind that so far it is unknown whether or not the additional structure brought by the defining polynomial in Ring-LWE can be exploited for cryptanalysis. Therefore the security estimations for Ring-LWE are made through analysis for an LWE instance.

Definition 1.1.40. Let $\mathcal{L}$ be a lattice of rank d in $\mathbb{R}^n$ and let $\mathcal{B}$ be a basis of $\mathcal{L}$. The *volume* of the lattice $\mathcal{L}$ is defined as $\sqrt{\det(B^T B)}$ where $B \in \mathcal{M}_{n,d}(\mathbb{R})$ is the matrix whose columns are formed by the elements of $\mathcal{B}$. When $\mathcal{L}$ is full-rank, it corresponds to $|\det(B)|$.

Remark 1.1.41. Two bases being related by a unimodular transformation, the determinant of a lattice does not depend on the chosen basis. It measures the volume of the d-dimensional polytope formed by the vectors of a basis.

Definition 1.1.42. For a lattice $\mathcal{L}$ of rank d, the *root-Hermite factor* $\gamma_\mathcal{B}$ associated to a basis $\mathcal{B}$ is defined as $\|b_1\| = \gamma_\mathcal{B}^d \det(\mathcal{L})^{1/d}$ where b_1 is the shortest vector of $\mathcal{B}$.

As mentioned in Remark 1.1.32, the best current strategy to solve S(I)VP or its approximate version is to use lattice reductions algorithms. We have already seen that lattices can be represented by a basis, however vectors appearing in bases can be of very different lengths. Intuitively a "good" basis is formed by short vectors almost orthogonal, i.e. with low orthogonality defect, by opposition a "bad" basis is formed by long vectors almost collinear. Lattice reduction algorithms aim at finding a "good" basis from an arbitrary basis, together with some guaranties on the norm and the orthogonality of the output. The most famous lattice reduction algorithm is probably the celebrated Lenstra Lenstra Lovász (LLL) ([LLL82]) which

runs in polynomial time and provides bases of descent quality. Another noteworthy lattice reduction algorithm is BKZ ([SE94]) which uses LLL and an SVP-oracle, usually based on enumeration techniques, on small blocks of $\beta < d$ vectors both as subroutine. For many cryptanalysis applications BKZ provides bases of higher quality in high dimension however its running time, as well as the quality of the output, increases significantly with the block size β. As a consequence, when considering the basis $\mathcal{B}'$ outputted by a lattice reduction algorithm $\mathcal{A}$, run on an input basis $\mathcal{B}$, we consider the value $\gamma_{\mathcal{A}(\mathcal{B})}$ to measure the quality of reduction offered by the algorithm:

$$\|b'_1\| = \gamma^d_{\mathcal{A}(\mathcal{B})} \det(\mathcal{L})^{1/d}$$

Gama and Nguyen ([GN08]) conjectured that $\gamma_{\mathcal{A}(\mathcal{B})}$ depends mostly on the lattice reduction algorithm used and not on the input basis, thus we will refer to this value as $\gamma_{\mathcal{A}}$. They also showed that when targeting a given factor $\gamma_{\mathcal{A}}$ in high dimension, the running time of the reduction algorithm mainly depends on $\gamma_{\mathcal{A}}$ and not on the dimension or the volume of the lattice. Thus, by following their idea, there is a minimal factor $\gamma^{min}_{\mathcal{A}}$ that one can achieve in time 2^{λ}, for security level of λ, with algorithm $\mathcal{A}$. In [LP11], Lindner and Peikert proposed a distinguishing attack which succeeds with probability ϵ very close to $\exp(-\pi(\|v\|\sigma_{err}/q)^2)$ with v a short vector of a lattice associated to the Ring-LWE instantiation with modulus q and error distribution χ_{err} whose standard deviation is σ_{err}. Therefore for the attack to succeed with probability ϵ one needs to find a vector of length at most $\alpha q/\sigma_{err}$ with $\alpha = \sqrt{-\log_2(\epsilon)/\pi}$. Furthermore they showed that for a lattice of rank n, the shortest length of vectors one can compute for a given $\gamma_{\mathcal{A}}$ is $2^{2\sqrt{n\log_2(q)\log_2(\gamma_{\mathcal{A}})}}$. As a consequence to ensure the security of a Ring-LWE instantiation the following inequality must hold:

$$\alpha\frac{q}{\sigma_{err}} < 2^{2\sqrt{n\log_2(q)\log_2(\gamma^{min}_{\mathcal{A}})}} \tag{1.1.3}$$

Many works were led to predict more accurately the behavior of BKZ, or other lattice reduction algorithms, to get better estimations of the root-Hermite factor one could reach in order to derive better security parameters. The interested reader can refer to the following non-exhaustive list ([CN11], [vdPS13], [LN14], [APS15], [MW16]). However inequality (1.1.3) allows us to understand the relations between the different parameters and the security, e.g. for a fixed dimension n, one cannot increase q beyond a certain bound without also increasing the standard deviation σ_{err} of the error distribution.

1.2 Somewhat Homomorphic Encryption based on Ring-LWE

This section aims to present two of the most currently used SHE schemes based on the Ring-LWE problem, namely BGV ([BGV12]) implemented in the Helib library ([HS14]) which is based on `NTL`, and Fan-Vercauteren (FV) ([FV12]) implemented in the SEAL library ([LP16], [CLP17]) developed by Microsoft Research. Ring-LWE based cryptography corresponds to

"noisy" protocols which means that data are hidden in cryptograms with some noise. Manipulating those cryptograms makes the noise grow and unfortunately data can only be got back as long as the noise remains smaller than a certain bound.

Throughout the rest of this thesis, $\mathbf{\Phi}_m(X) \in \mathbb{Z}[X]$ will denote the m-th cyclotomic polynomial of degree $n = \varphi(m)$, where φ is Euler's totient function. The ring $\mathcal{R} = \mathbb{Z}[X]/(\mathbf{\Phi}_m(X))$ is the main structure of Ring-LWE-based schemes such as `FV` and `BGV`. An element of $\mathcal{R}$ can be thought of as a polynomial with integer coefficients and a degree strictly smaller than n. Unless mentioned otherwise, polynomials are represented in the power-basis $\{1, X, \dots X^{n-1}\}$ and are denoted with bold lower-case letters. For $\boldsymbol{a} = \sum_{i=0}^{n-1} a_i X^i \in \mathbb{Z}[X]$, we denote $\|\boldsymbol{a}\|_\infty = \max\{|a_i|, 0 \leqslant i < n\}$ the infinite norm on the coefficients. The *expansion factor* of the ring $\mathcal{R}$ is defined by $\delta_\mathcal{R} = \sup\{\|\boldsymbol{a} \cdot \boldsymbol{b}\|_\infty / \|\boldsymbol{a}\|_\infty \cdot \|\boldsymbol{b}\|_\infty$ for $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{R} - \{0\}\}$ and quantifies the size growth of the coefficients during a multiplication in $\mathcal{R}$ in the worst-case scenario. The underlying space for ciphertexts is $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} = (\mathbb{Z}/q\mathbb{Z})[X]/(\phi_m(X))$, which is composed of elements of $\mathcal{R}$ with coefficients reduced modulo q while the plaintext space is $\mathcal{R}_t$ for an integer $t \ll q$.

The notation $|\cdot|_q$ is used to denote the classical residue modulo q in $[0, q)$ of an integer, while the centered residue in $[-q/2, q/2)$ is denoted by $[\cdot]_q$. Moreover $\lfloor \cdot \rfloor$ denotes flooring while $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. By applying them coefficient-wise, those notations can be extended to polynomials.

The following two functions are needed to limit the noise growth during an homomorphic multiplication in the `BGV` and `FV` schemes. They are applicable to any $\boldsymbol{a} \in \mathcal{R}$, for any radix $\omega \geq 2$, and with the subsequent parameter $\ell_{\omega,q} = \lfloor \log_\omega(q) \rfloor + 1$. $\mathcal{D}_{\omega,q}$ is a decomposition in radix base ω, while $\mathcal{P}_{\omega,q}$ retrieves powers of ω which are lost within the decomposition process.

$$\mathcal{D}_{\omega,q}(\boldsymbol{a}) = \left([\boldsymbol{a}]_\omega, \left[\left\lfloor \frac{\boldsymbol{a}}{\omega} \right\rfloor \right]_\omega, \dots, \left[\left\lfloor \frac{\boldsymbol{a}}{\omega^{\ell_{\omega,q}-1}} \right\rfloor \right]_\omega \right) \in \mathcal{R}_\omega^{\ell_{\omega,q}}$$

$$\mathcal{P}_{\omega,q}(\boldsymbol{a}) = \left([\boldsymbol{a}]_q, [\boldsymbol{a}\omega]_q, \dots, [\boldsymbol{a}\omega^{\ell_{\omega,q}-1}]_q \right) \in \mathcal{R}_q^{\ell_{\omega,q}}$$

Lemma 1.2.1 ([BGV12] lemma 2)**.** For any $(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}^2$, $\left\langle \mathcal{D}_{\omega,q}(\boldsymbol{a}), \mathcal{P}_{\omega,q}(\boldsymbol{b}) \right\rangle \equiv \boldsymbol{a} \cdot \boldsymbol{b} \bmod q$.

$\mathcal{U}_q$ will denote the uniform distribution on $\mathcal{R}_q$ with coefficients sampled independently in $\mathbb{Z} \cap [-q/2, q/2)$. We will also consider two distributions χ_{key} and χ_{err} on $\mathcal{R}$ which are supposed bounded by B_{key} and B_{err} respectively. It means that any $\boldsymbol{a}$ sampled randomly from χ_{key} (resp. χ_{err}) verifies $\|\boldsymbol{a}\|_\infty \leq B_{key}$ (resp. B_{err}). The distribution χ_{err} corresponds to a distribution statistically indistinguishable from a discrete centered Gaussian of standard deviation σ_{err} truncated at B_{err}, for a large enough B_{err} (e.g. $B_{err} = 6\sigma_{err}$). Despite Ring-LWE problems being not easier when the secret is sampled from the error distribution χ_{err} ([LPR10]), the key distribution χ_{key} is often chosen different than χ_{err} (e.g. uniform in $\{-1, 0, 1\}$). Since noise growth depends on the size of the secret key in many situations, this leads to more efficient protocols. However, such choice might introduce security weaknesses, and thus would require further investigations for any commercial deployment.

1.2.1 Brakerski-Gentry-Vaikuntanathan

In 2011, Brakerski et al. ([BGV12]) designed a leveled homomorphic scheme, namely capable of evaluating circuits of arbitrary size, but known beforehand, without involving the costly bootstrapping procedure of Gentry. The key tool of their construction is the *modulus switching* procedure which allows to switch a ciphertext $\texttt{ct}$ encrypted under a modulus q to a smaller modulus q' in order to keep the noise level "constant". Hence by selecting a chain of moduli $\{q_0, \dots, q_L\}$ long enough to perform the desired computations, bootstrapping is no longer needed. This section describes the $\texttt{BGV}$ encryption scheme in the Ring-LWE setting.

First, accordingly to the desired security level and the targeted application one starts by selecting public parameters $\texttt{params}_{\texttt{BGV}}$ for the scheme: the cyclotomic index m, the plaintext modulus t, an radix $\omega \geq 2$, two probability distributions χ_{key} and χ_{err} on the ring $\mathcal{R}$, and a chain of moduli $q_0 \mid q_1 \mid \dots \mid q_L$ such that $q_i \equiv q_j \bmod t$ for any $(i,j) \in \{0,\dots,L\}^2$, and such that t and q_L are coprime.

$$\texttt{params}_{\texttt{BGV}} = (m, t, \omega, \{q_i\}_{i=0}^{L}, \chi_{key}, \chi_{err})$$

Remark 1.2.2. Usually q_L is a product of $L+1$ primes of same size and t is a power of two.

Key generation. The secret key $\texttt{sk} = \boldsymbol{s} \in \mathcal{R}$ is sampled randomly from the distribution χ_{key}. The public key $\texttt{pk} = ([\boldsymbol{a}\cdot\boldsymbol{s} + t\boldsymbol{e}]_{q_L}, -\boldsymbol{a}) \in \mathcal{R}_{q_L}^2$ corresponds roughly to a Ring-LWE sample associated to $\boldsymbol{s}$ and q_L with $\boldsymbol{a} \xleftarrow{\$} \mathcal{U}_{q_L}$ and $\boldsymbol{e} \xleftarrow{\$} \chi_{err}$. Finally one generates a public relinearization key $\overrightarrow{\texttt{rlk}} = ([\mathcal{P}_{\omega,q}(\boldsymbol{s}^2) + \vec{\boldsymbol{a}}\cdot\boldsymbol{s} + t\vec{\boldsymbol{e}}]_{q_L}, -\vec{\boldsymbol{a}}) \in \mathcal{R}_{q_L}^{\ell_{\omega,q_L}} \times \mathcal{R}_{q_L}^{\ell_{\omega,q_L}}$ by sampling randomly a vector $\vec{\boldsymbol{a}} \in \mathcal{R}_{q_L}^{\ell_{\omega,q_L}}$ (resp. $\vec{\boldsymbol{e}} \in \mathcal{R}_{q_L}^{\ell_{\omega,q_L}}$) with each component sampled independently from $\mathcal{U}_{q_L}$ (resp χ_{err}). Note that $\boldsymbol{s}$ is multiplied to each component of $\vec{\boldsymbol{a}}$.

Remark 1.2.3. One can think to $\overrightarrow{\texttt{rlk}}$ as an encryption of $\boldsymbol{s}^2$.

Algorithm 1 Key Generation BGV

Require: the public parameters $\texttt{params}_{\texttt{BGV}}$.
Ensure: secret key $\texttt{sk}$, public key $\texttt{pk}$ and relinearization key $\overrightarrow{\texttt{rlk}}$

function KeyGen$_{\texttt{BGV}}$($\texttt{params}_{\texttt{BGV}}$)
- $\boldsymbol{s} \xleftarrow{\$} \chi_{key}$
- $\texttt{sk} \leftarrow \boldsymbol{s}$
- $(\boldsymbol{a}, \boldsymbol{e}) \xleftarrow{\$} \mathcal{U}_{q_L} \times \chi_{err}$
- $\texttt{pk} \leftarrow ([\boldsymbol{a}\cdot\boldsymbol{s} + t\boldsymbol{e}]_{q_L}, -\boldsymbol{a})$
- $(\vec{\boldsymbol{a}}, \vec{\boldsymbol{e}}) \xleftarrow{\$} \mathcal{U}_{q_L}^{\ell_{\omega,q_L}} \times \chi_{err}^{\ell_{\omega,q_L}}$
- $\overrightarrow{\texttt{rlk}} \leftarrow \left(\left[\mathcal{P}_{\omega,q_L}(\boldsymbol{s}^2) + \vec{\boldsymbol{a}}\cdot\boldsymbol{s} + t\vec{\boldsymbol{e}}\right]_{q_L}, -\vec{\boldsymbol{a}}\right)$
- **return** $(\texttt{sk}, \texttt{pk}, \overrightarrow{\texttt{rlk}})$

Encryption A ciphertext $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_{q_L}^2$ corresponds to a degree 1 polynomial whose coefficients lie in $\mathcal{R}_{q_L}$. The message $\boldsymbol{m} \in \mathcal{R}_t$ is hidden in the first coefficient $\boldsymbol{c}_0$ of the ciphertext, more precisely a message $\boldsymbol{m}$ is encrypted under the public key $\mathtt{pk}$ as follows:

$$\mathtt{ct} = \left([[\boldsymbol{m}]_t + \boldsymbol{u} \cdot \mathbf{pk}_0 + t\boldsymbol{e}_0]_{q_L}, [\boldsymbol{u} \cdot \mathbf{pk}_1 + t\boldsymbol{e}_1]_{q_L}\right)$$

with $\boldsymbol{u} \xleftarrow{\$} \chi_{key}$ and $\boldsymbol{e}_0, \boldsymbol{e}_1 \xleftarrow{\$} \chi_{err}$. The noise contained in a ciphertext $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ is revealed once it is evaluated on the secret key $\boldsymbol{s}$, indeed the following equalities hold modulo q_L:

$$\begin{aligned}
\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} &= [\boldsymbol{m}]_t + \boldsymbol{u} \cdot \mathbf{pk}_0 + t\boldsymbol{e}_0 + (\boldsymbol{u} \cdot \mathbf{pk}_1 + t\boldsymbol{e}_1) \cdot \boldsymbol{s} \\
&= [\boldsymbol{m}]_t + \boldsymbol{u} \cdot (\boldsymbol{a} \cdot \boldsymbol{s} + t\boldsymbol{e}) + t\boldsymbol{e}_0 + (-\boldsymbol{u} \cdot \boldsymbol{a} + t\boldsymbol{e}_1) \cdot \boldsymbol{s} \\
&= [\boldsymbol{m}]_t + t(\boldsymbol{u} \cdot \boldsymbol{e} + \boldsymbol{e}_1 \cdot \boldsymbol{s} + \boldsymbol{e}_0)
\end{aligned}$$

which leads to:

$$\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} \equiv [\boldsymbol{m}]_t + t\boldsymbol{v} \bmod q_L \tag{1.2.1}$$

where the term $\boldsymbol{v} = \boldsymbol{u} \cdot \boldsymbol{e} + \boldsymbol{e}_1 \cdot \boldsymbol{s} + \boldsymbol{e}_0$ is the noise inherent to a ciphertext "freshly" encrypted. Since $q_0 \mid q_1 \mid \ldots \mid q_L$, encryptions can be performed equivalently at any level i, i.e. modulo q_i therefore ciphertexts are always paired with a number $0 \leq i \leq L$ which indicates the level of the encryption.

Algorithm 2 Encryption BGV

Require: message $\boldsymbol{m} \in \mathcal{R}_t$, public key $\mathtt{pk} = (\mathbf{pk}_0, \mathbf{pk}_1)$ and the public parameters $\mathtt{params}_{\mathtt{BGV}}$.
Ensure: level i ciphertext $\mathtt{ct} = ((\boldsymbol{c}_0, \boldsymbol{c}_1), i)$ encrypting $\boldsymbol{m}$.

function $\textsc{Enc}_{\mathtt{BGV}}(\boldsymbol{m}, \mathtt{pk}, \mathtt{params}_{\mathtt{BGV}})$
- $\boldsymbol{u} \xleftarrow{\$} \chi_{key}$
- $(\boldsymbol{e}_0,\ \boldsymbol{e}_1) \xleftarrow{\$} \chi_{err}^2$
- **return** $\mathtt{ct} = \left(\left([[\boldsymbol{m}]_t + \boldsymbol{u} \cdot \mathbf{pk}_0 + t\boldsymbol{e}_0]_{q_i}, [\boldsymbol{u} \cdot \mathbf{pk}_1 + t\boldsymbol{e}_1]_{q_i}\right), i\right)$

Decryption As shown in equation (1.2.1), by evaluating a ciphertext on the secret key one almost reveals the message. Therefore to decrypt a message $\boldsymbol{m} \in \mathcal{R}_t$ encrypted at level i in a ciphertext $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_{q_i}^2$, one starts by computing $\boldsymbol{m}' = [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_i}$ and then outputs $[\boldsymbol{m}']_t$. To ensure correctness of the decryption, the noise $\boldsymbol{v}$ must be "small enough" such that $\boldsymbol{m}' = \boldsymbol{m} + t\boldsymbol{v}$ does not wrap-around modulo q_i. As a consequence decryption will remain correct if:

$$\|[\boldsymbol{m}]_t + t\boldsymbol{v}\|_\infty < \frac{q_i}{2}$$

which is ensured as long as:

$$\|\boldsymbol{v}\|_\infty < \left\lfloor \frac{q_i}{2t} \right\rfloor \tag{1.2.2}$$

As explained above, the noise of fresh ciphertexts is $\boldsymbol{v} = \boldsymbol{u} \cdot \boldsymbol{e} + \boldsymbol{e}_1 \cdot \boldsymbol{s} + \boldsymbol{e}_0$ with $\boldsymbol{s}, \boldsymbol{u} \xleftarrow{\$} \chi_{key}$ and $\boldsymbol{e}, \boldsymbol{e}_0, \boldsymbol{e}_1 \xleftarrow{\$} \chi_{err}$. Therefore one way to ensure the correctness of the scheme is to choose q_0 such that the norm of this fresh noise $\|\boldsymbol{v}\|_\infty \leq 2\delta_{\mathcal{R}} B_{key} B_{err} + B_{err}$ remains smaller than $\lfloor q_0/2t \rfloor$ i.e. $B_{err}(2\delta_{\mathcal{R}} B_{key} + 1) < \lfloor q_0/2t \rfloor$, with $\delta_{\mathcal{R}}$ the expansion factor of the ring $\mathcal{R}$.

Algorithm 3 Decryption BGV

Require: ciphertext $\mathtt{ct} = ((\boldsymbol{c}_0, \boldsymbol{c}_1), i)$, secret key $\mathtt{sk} = \boldsymbol{s}$ and the public parameters $\mathtt{params}_{\mathtt{BGV}}$.
Ensure: message $[\boldsymbol{m}]_t$ (if equation (1.2.2) is satisfied).
  function $\textsc{Dec}_{\mathtt{BGV}}(\mathtt{ct}, \mathtt{sk}, \mathtt{params}_{\mathtt{BGV}})$
    $\boldsymbol{m}' \leftarrow [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_i}$
    return $[\boldsymbol{m}']_t$

Modulus switching The modulus switching is a noise management technique allowing to decrease the size of the noise of a level $j > 0$ ciphertext, as soon as it becomes too important (e.g. $\|\boldsymbol{v}\|_\infty > \lfloor q_j/2t \rfloor$). Roughly, the idea is to drop one (or several) levels in the ladder of moduli by scaling the ciphertext by q_i/q_j for $i < j$, which roughly scales down the noise by the same factor. More precisely, let $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ be a level $j \in (0, L] \cap \mathbb{Z}$ encryption of a message $\boldsymbol{m}$ and let i be an integer smaller than j, then set:

$$\boldsymbol{\delta} = (t[-\boldsymbol{c}_0/t]_{q_j/q_i}, t[-\boldsymbol{c}_1/t]_{q_j/q_i})$$

In this way for $l = 0, 1$, $\boldsymbol{\delta}_l \equiv 0 \bmod t$ and $\boldsymbol{\delta}_l \equiv -\boldsymbol{c}_l \bmod q_j/q_i$ and we can compute:

$$\mathtt{ct}' = \frac{q_i}{q_j} \cdot (\boldsymbol{c}_0 + \boldsymbol{\delta}_0, \boldsymbol{c}_1 + \boldsymbol{\delta}_1)$$

Since $\boldsymbol{c}_l + \boldsymbol{\delta}_l \equiv 0 \bmod q_j/q_i$, the division by q_j/q_i is exact. Now if we want $\mathtt{ct}' = (\boldsymbol{c}'_0, \boldsymbol{c}'_1)$ to be a level i encryption of $\boldsymbol{m}$, we need to ensure $[\boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s}]_{q_i} = [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} \bmod t$. Let $\boldsymbol{k}$ be such that:

$$[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} = \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} - q_j \boldsymbol{k}$$

and so we have:

$$\begin{aligned} \boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s} &= \frac{q_i}{q_j}(\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} + \boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s}) \\ \Leftrightarrow \boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s} - q_i \boldsymbol{k} &= \frac{q_i}{q_j}\left([\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} + \boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s}\right) \end{aligned}$$

notice that for $l = 0, 1$ $\|\boldsymbol{\delta}_l\|_\infty < \frac{t q_j}{2 q_i}$, it implies:

$$\|\boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s}\|_\infty \leq \frac{t q_j}{2 q_i}(1 + \delta_{\mathcal{R}} B_{key})$$

therefore if $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ is such that:

$$\left\| [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} \right\|_\infty < \frac{q_j}{2} - \frac{t q_j}{2 q_i}(1 + \delta_{\mathcal{R}} B_{key}) \tag{1.2.3}$$

we have:

$$\left\| \boldsymbol{c}_0' + \boldsymbol{c}_1' \cdot \boldsymbol{s} - q_i \boldsymbol{k} \right\|_\infty < \frac{q_i}{2}$$

and so we obtain what we want:

$$\begin{aligned} [\boldsymbol{c}_0' + \boldsymbol{c}_1' \cdot \boldsymbol{s}]_{q_i} &= \boldsymbol{c}_0' + \boldsymbol{c}_1' \cdot \boldsymbol{s} - q_i \boldsymbol{k} \\ &= \frac{q_i}{q_j}([\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} + \boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s}) \\ &= [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} \bmod t \quad (\boldsymbol{\delta}_l \equiv 0 \bmod t \text{ and } q_j \equiv q_i \bmod t) \end{aligned}$$

Furthermore the noise $\boldsymbol{v}'$ of $\mathtt{ct}'$ is given by:

$$\boldsymbol{v}' = \frac{q_i}{q_j}\boldsymbol{v} + \boldsymbol{b}_{\text{scale}} \tag{1.2.4}$$

with:

$$\|\boldsymbol{b}_{\text{scale}}\|_\infty = \left\| \frac{q_i}{t q_j}(\boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s}) \right\|_\infty \leq \frac{1 + \delta_{\mathcal{R}} B_{key}}{2}$$

Therefore when a level i ciphertext satisfies inequality (1.2.3), one is able to reduce the noise of this ciphertext by converting it to a level $i < j$. In particular, once a ciphertext reaches level 0, this method cannot be applied anymore.

Algorithm 4 Modulus Switching BGV

Require: ciphertext $\mathtt{ct} = ((\boldsymbol{c}_0, \boldsymbol{c}_1), i)$ encrypted at level $j > 0$ satisfying equation (1.2.3), a level $i < j$ and the public parameters $\mathtt{params}_{\mathtt{BGV}}$.
Ensure: ciphertext $\mathtt{ct}' = ((\boldsymbol{c}_0', \boldsymbol{c}_1'), i)$ encrypting the same message than $\mathtt{ct}$ at level i
function MOD-SWITCH$_{\mathtt{BGV}}(\mathtt{ct}, j, \mathtt{params}_{\mathtt{BGV}})$
$\boldsymbol{\delta} \leftarrow (t[-\boldsymbol{c}_0/t]_{q_j/q_i}, t[-\boldsymbol{c}_1/t]_{q_j/q_i})$
$\mathtt{ct}' \leftarrow \left(\left(\left[\frac{q_i}{q_j}(\boldsymbol{c}_0 + \boldsymbol{\delta}_0) \right]_{q_i}, \left[\frac{q_i}{q_j}(\boldsymbol{c}_1 + \boldsymbol{\delta}_1) \right]_{q_i} \right), j \right)$
return $\mathtt{ct}'$

Addition In the case where $\mathtt{ct}$ is encrypted at level i and $\mathtt{ct}'$ at level $j > i$, then one starts by applying modulus switching to $\mathtt{ct}$ from level j to level i. Once both at same level i, adding the two ciphertexts, which encrypt $\boldsymbol{m}$ and $\boldsymbol{m}'$ respectively, leads to:

$$\begin{aligned} \boldsymbol{c}_0 + \boldsymbol{c}_0' + (\boldsymbol{c}_1 + \boldsymbol{c}_1') \cdot \boldsymbol{s} = \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} + \boldsymbol{c}_0' + \boldsymbol{c}_1' \cdot \boldsymbol{s} &\equiv [\boldsymbol{m}]_t + [\boldsymbol{m}']_t + t(\boldsymbol{v} + \boldsymbol{v}') \bmod q_i \\ &\equiv [\boldsymbol{m} + \boldsymbol{m}']_t + t(\boldsymbol{v} + \boldsymbol{v}' + \boldsymbol{u}) \bmod q_i \end{aligned}$$

where $[\boldsymbol{m}]_t + [\boldsymbol{m}']_t = [\boldsymbol{m} \cdot \boldsymbol{m}']_t + \boldsymbol{u}$ with $\|\boldsymbol{u}\|_\infty \leq 1$. This means that:

$$\mathtt{ct}_{\text{add}} = \left([\boldsymbol{c}_0 + \boldsymbol{c}'_0]_{q_i}, [[\boldsymbol{c}_1 + \boldsymbol{c}'_1]_{q_i}]\right)$$

is a level i encryption of $\boldsymbol{m} + \boldsymbol{m}'$ and its noise is roughly the sum of the noises of $\mathtt{ct}$ and $\mathtt{ct}'$:

$$\boldsymbol{v}_{\text{add}} = \boldsymbol{v} + \boldsymbol{v}' + \boldsymbol{u}.$$

of norm bounded by:

$$\|\boldsymbol{v}_{\text{add}}\|_\infty \leq \|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}\|_\infty + 1. \tag{1.2.5}$$

Algorithm 5 Addition BGV

Require: ciphertexts $\mathtt{ct} = ((\boldsymbol{c}_0, \boldsymbol{c}_1), i)$ and $\mathtt{ct}' = ((\boldsymbol{c}'_0, \boldsymbol{c}'_1), j)$ encrypting messages $\boldsymbol{m}$ and $\boldsymbol{m}'$ respectively and the public parameters $\mathtt{params}_{\mathtt{BGV}}$.
Ensure: ciphertext $\mathtt{ct}_{\text{add}}$ encrypting $\boldsymbol{m} + \boldsymbol{m}'$ at level $k = \min(i, j)$.

```
function ADD_BGV(ct, ct', params_BGV)
    if i ≠ j then
        if i < j then
            ct̃' ← MOD-SWITCH_BGV(ct', i, params_BGV)
            ct̃ ← ct
        else
            ct̃ ← MOD-SWITCH_BGV(ct, j, params_BGV)
            ct̃' ← ct'
    k ← min(i, j)
    ct_add ← ([c̃_0 + c̃_0']_{q_k}, [c̃_1 + c̃_1']_{q_k}, k)
    return ct_add
```

Multiplication Similarly to the addition, we first need to convert the two ciphertexts to the same level, let say i, by applying the modulus switching technique. Then multiplying the two ciphertexts leads to:

$$\begin{aligned}
\boldsymbol{c}_0 \cdot \boldsymbol{c}'_0 + (\boldsymbol{c}_0 \cdot \boldsymbol{c}'_1 + \boldsymbol{c}_1 \cdot \boldsymbol{c}'_0) \cdot \boldsymbol{s} + \boldsymbol{c}_0 \cdot \boldsymbol{c}'_0 \cdot \boldsymbol{s}^2 &= (\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}) \cdot (\boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s}) \\
&\equiv [\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t + t([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v}' \cdot [\boldsymbol{m}']_t + t\boldsymbol{v} \cdot \boldsymbol{v}') \bmod q_i \\
&\equiv [\boldsymbol{m} \cdot \boldsymbol{m}']_t + t([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v}' \cdot [\boldsymbol{m}']_t + t\boldsymbol{v} \cdot \boldsymbol{v}' + \boldsymbol{r}_m) \bmod q_i
\end{aligned}$$

with $[\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t = [\boldsymbol{m} \cdot \boldsymbol{m}']_t + t\boldsymbol{r}_m$ which implies that:

$$\|\boldsymbol{r}_m\|_\infty = \left\| \frac{[\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t - [\boldsymbol{m} \cdot \boldsymbol{m}']_t}{t} \right\|_\infty \leq \frac{\delta_{\mathcal{R}} t^2 + 2t}{4t} \leq \frac{\delta_{\mathcal{R}} t}{4} + \frac{1}{2} < \frac{\delta_{\mathcal{R}} t}{2} \tag{1.2.6}$$

This means that:

$$\mathtt{ct}_{\text{mult}} = ([\boldsymbol{c}_0 \cdot \boldsymbol{c}'_0]_{q_i}, [\boldsymbol{c}_0 \cdot \boldsymbol{c}'_1 + \boldsymbol{c}_1 \cdot \boldsymbol{c}'_0]_{q_i}, [\boldsymbol{c}_1 \cdot \boldsymbol{c}'_1]_{q_i}) \in \mathcal{R}^3_{q_i} \tag{1.2.7}$$

is a degree 2 ciphertext which encrypts $\boldsymbol{m} \cdot \boldsymbol{m}'$ and its noise is given by:

$$\boldsymbol{v}_{\text{mult}} = [\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v}' \cdot [\boldsymbol{m}']_t + t\boldsymbol{v} \cdot \boldsymbol{v}' + \boldsymbol{r}_m$$

whose norm is bounded by:

$$\|\boldsymbol{v}_{\text{mult}}\|_\infty < \frac{\delta_{\mathcal{R}} t}{2} \left(\|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}'\|_\infty + \|\boldsymbol{v}\|_\infty \|\boldsymbol{v}'\|_\infty + 1\right) \tag{1.2.8}$$

Algorithm 6 Multiplication BGV

Require: ciphertexts $\texttt{ct} = ((\boldsymbol{c}_0, \boldsymbol{c}_1), i)$ and $\texttt{ct}' = ((\boldsymbol{c}_0', \boldsymbol{c}_1'), j)$ encrypting messages $\boldsymbol{m}$ and $\boldsymbol{m}'$ respectively and the public parameters $\texttt{params}_{\texttt{BGV}}$.
Ensure: degree 2 ciphertext $\texttt{ct}_{\text{mult}}$ encrypting $\boldsymbol{m} \cdot \boldsymbol{m}'$ at level $k = \min(i, j)$.

```
function MULT_BGV(ct, ct', params_BGV)
    if i ≠ j then
        if i < j then
            ct̃' ← MOD-SWITCH_BGV(ct', i, params_BGV)
            ct̃ ← ct
        else
            ct̃ ← MOD-SWITCH_BGV(ct, j, params_BGV)
            ct̃' ← ct'
    k ← min(i, j)
    ct_mult ← ([c̃_0 · c̃_0']_{q_k}, [c̃_0 · c̃_1' + c̃_1 · c̃_0']_{q_k}, [c̃_1 · c̃_1']_{q_k}, k)
    return ct_mult
```

Relinearization The drawback of the multiplication is the growth of the ciphertext's degree in the same way as in polynomials. Therefore after several multiplications the complexity in terms of time and memory for further operations is considerably increased. In order to keep the ciphertexts' degree constant, one can use a relinearization technique, introduced in [BV11], or sometimes called *key switching* when ciphertexts reach degree 2.

A degree 2 ciphertext $\texttt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2)$ which encrypts a message $\boldsymbol{m}$ at level i, satisfies:

$$\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} + \boldsymbol{c}_2 \cdot \boldsymbol{s}^2 \equiv [\boldsymbol{m}]_t + t\boldsymbol{v} \bmod q_i$$

Essentially the idea would be to add $\boldsymbol{c}_2 \cdot \boldsymbol{s}^2$ to $\boldsymbol{c}_0$ to get back a degree 1 ciphertext. Although $\boldsymbol{s}^2$ cannot be published, it can be hidden in a kind of encryption under the public key $\texttt{pk}$. This is exactly what is done in $\overrightarrow{\texttt{rlk}}$. However since $\boldsymbol{s}^2$ is encrypted with $\texttt{pk}$ which is derived from $\boldsymbol{s}$ it requires to use a circular security assumption, i.e. to assume that encrypting a secret key with its corresponding public key remains safe. Hence by computing:

$$\tilde{\texttt{ct}} = \left(\left[\boldsymbol{c}_0 + \left\langle \mathcal{D}_{\omega, q_i}(\boldsymbol{c}_2), \overrightarrow{\texttt{rlk}}_0 \right\rangle \right]_{q_i}, \left[\boldsymbol{c}_1 + \left\langle \mathcal{D}_{\omega, q_i}(\boldsymbol{c}_2), \overrightarrow{\texttt{rlk}}_1 \right\rangle \right]_{q_i} \right) \in \mathcal{R}_{q_i}^2$$

we obtain a degree 1 encryption of $\boldsymbol{m}$ at level i, indeed:

$$\begin{aligned}
\tilde{\boldsymbol{c}}_0 + \tilde{\boldsymbol{c}}_1 \boldsymbol{s} &\equiv \boldsymbol{c}_0 + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_0} \right\rangle + \left(\boldsymbol{c}_1 + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_1} \right\rangle \right) \cdot \boldsymbol{s} \bmod q_i \\
&\equiv \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_0} \right\rangle + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_1} \right\rangle \cdot \boldsymbol{s} \bmod q_i \\
&\equiv \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \mathcal{P}_{\omega,q_i}(\boldsymbol{s}^2) + \vec{\boldsymbol{a}} \cdot \boldsymbol{s} + t\vec{\boldsymbol{e}} \right\rangle - \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \vec{\boldsymbol{a}} \right\rangle \cdot \boldsymbol{s} \bmod q_i \\
&\equiv \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} + \boldsymbol{c}_2 \cdot \boldsymbol{s}^2 + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \vec{\boldsymbol{a}} \cdot \boldsymbol{s} + t\vec{\boldsymbol{e}} \right\rangle - \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \vec{\boldsymbol{a}} \right\rangle \cdot \boldsymbol{s} \bmod q_i \\
&\equiv \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} + \boldsymbol{c}_2 \cdot \boldsymbol{s}^2 + t \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \vec{\boldsymbol{e}} \right\rangle \bmod q_i \\
&\equiv [\boldsymbol{m}]_t + t(\boldsymbol{v} + \boldsymbol{b}_{\text{relin}}) \bmod q_i
\end{aligned}$$

with a noise given by:

$$\boldsymbol{v}_{\text{relin}} = \boldsymbol{v} + \boldsymbol{b}_{\text{relin}} = \boldsymbol{v} + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \vec{\boldsymbol{e}} \right\rangle$$

whose norm is bounded by:

$$\begin{aligned}
\|\boldsymbol{v}_{\text{relin}}\|_\infty &\leq \|\boldsymbol{v}\|_\infty + \|\boldsymbol{b}_{\text{relin}}\|_\infty \\
&\leq \|\boldsymbol{v}\|_\infty + \left\| \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \vec{\boldsymbol{e}} \right\rangle \right\|_\infty \\
&\leq \|\boldsymbol{v}\|_\infty + \sum_{j=0}^{\ell_{\omega,q_i}-1} \left\| \left[\left\lfloor \frac{\boldsymbol{c}_2}{\omega^j} \right\rceil \right]_\omega \cdot \vec{\boldsymbol{e}}_j \right\|_\infty \\
&\leq \|\boldsymbol{v}\|_\infty + \frac{\delta_{\mathcal{R}} \ell_{\omega,q_i} \omega B_{err}}{2}
\end{aligned} \tag{1.2.9}$$

Remark 1.2.4. $\boldsymbol{c}_2$ is decomposed in radix ω because otherwise there would have been a factor q_i instead of $\ell_{\omega,q_i}\omega$ ($\simeq \log q_i$) in the right term of (1.2.9). In which case the noise caused by the relinearization would be potentially much bigger than the decryption bound given in equation (1.2.2) which cannot be tolerated.

Algorithm 7 Relinearization BGV

Require: degree 2 ciphertext $\mathtt{ct} = ((\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2), i)$, the relinearization key $\overrightarrow{\mathtt{rlk}} = (\overrightarrow{\mathtt{rlk}_0}, \overrightarrow{\mathtt{rlk}_1})$ and the public parameters $\mathtt{params}_{\mathtt{BGV}}$.
Ensure: degree 1 ciphertext $\mathtt{ct}' = ((\boldsymbol{c}'_0, \boldsymbol{c}'_1), i)$ encrypting the same message than $\mathtt{ct}$.
function RELINEARIZATION$_{\mathtt{BGV}}$($\mathtt{ct}, \overrightarrow{\mathtt{rlk}}, \mathtt{params}_{\mathtt{BGV}}$)
$\mathtt{ct}' \leftarrow \left(\left(\left[\boldsymbol{c}_0 + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_0} \right\rangle \right]_{q_i}, \left[\boldsymbol{c}_1 + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_1} \right\rangle \right]_{q_i} \right), i \right)$
return $\mathtt{ct}'$

Parameters selection As shown in equation (1.2.2), in order to correctly decrypt the noise must remain smaller than $\lfloor q_0/2t \rfloor$. The noise growth during additions is linear (eq. (1.2.15)) and thus quite slow. Hence, it is not necessary to perform a modulus switching technique after each addition. However noise growth during multiplication is quadratic (eq. (1.2.8)), actually it is even more important since each multiplication is followed by a relinearization. Therefore,

to prevent the noise to blow up, it is necessary to run a modulus switching procedure at the end of the multiplication plus relinearization procedures. In order to be able to evaluate arithmetic circuits of multiplicative depth L, we must ensure that the noise, after being scaled down by one level, at the end of the whole procedure remains smaller than the noise of the input ciphertexts. Let us assume that the two level i ciphertexts in input of the multiplication have a noise whose norm is bounded by V ($V < \lfloor q_0/2t \rfloor$). By combining inequalities (1.2.8), (1.2.9) and (1.2.4) we can show that the noise of the level $i-1$ ciphertext at the end of the three procedures has norm bounded by:

$$\|\hat{\boldsymbol{v}}\|_\infty \leq \frac{q_{i-1}}{q_i}\frac{\delta_{\mathcal{R}} t}{2}\left(V^2 + 2V + 1\right) + \|\boldsymbol{b}_{\text{relin}}\|_\infty + \|\boldsymbol{b}_{\text{scale}}\|_\infty$$

Therefore if the parameters are selected such that:

$$\begin{cases} \dfrac{q_i}{q_{i-1}} \geq \delta_{\mathcal{R}} t V \text{ for all } 1 \leq i \leq L \\ 2 + \|\boldsymbol{b}_{\text{relin}}\|_\infty + \|\boldsymbol{b}_{\text{scale}}\|_\infty \leq \dfrac{V}{2} \end{cases} \tag{1.2.10}$$

we obtain:

$$\begin{aligned} \|\hat{\boldsymbol{v}}\|_\infty &\leq \frac{1}{2V}\left(V^2 + 2V + 1\right) + \|\boldsymbol{b}_{\text{relin}}\|_\infty + \|\boldsymbol{b}_{\text{scale}}\|_\infty \\ &\leq \frac{V}{2} + 2 + \|\boldsymbol{b}_{\text{relin}}\|_\infty + \|\boldsymbol{b}_{\text{scale}}\|_\infty \\ &\leq V \end{aligned}$$

which means that for a chain of $L+1$ moduli, we are able to evaluate circuits of multiplicative depth L.

1.2.2 Fan-Vercauteren

The main drawback of schemes such as `BGV` is the quadratic growth of the noise during multiplication which requires the use of the modulus switching technique. In its seminal work [Bra12], Brakerski introduced a new type of homomorphic schemes where the noise grows only linearly during multiplication removing thereby the necessity of modulus switching. Therefore all the computations require only a single modulus q by opposition to the chain of moduli in `BGV`. This is achieved by placing the message in the "upper bits", as opposed to the "lower bits", of the decryption equation (1.2.1). This more effective noise control mechanism makes the *scale-invariant* schemes, as they are called, particularly interesting. In 2012, Fan and Vercauteren [FV12] have adapted the scale-invariant scheme of Brakerski ([Bra12]) to the Ring-LWE setting. We present their scheme (`FV`) in this section.

Like for `BGV`, one starts by selecting public parameters $\texttt{params}_{\texttt{FV}}$ according to the desired security level and the targeted application. These parameters are: the cyclotomic index m, the plaintext modulus t, a radix $\omega \geq 2$, the ciphertext modulus $q \gg t$ and two probability

distributions χ_{key} and χ_{err} on the ring $\mathcal{R}$.

$$\texttt{params}_{\texttt{FV}} = (m, t, \omega, q, \chi_{key}, \chi_{err})$$

Key generation. The secret key $\texttt{sk} = \boldsymbol{s} \in \mathcal{R}$ is sampled randomly from the distribution χ_{key}. The public key $\texttt{pk} = ([\boldsymbol{a}\cdot\boldsymbol{s}+\boldsymbol{e}]_q, -\boldsymbol{a}) \in \mathcal{R}_q^2$ corresponds this time exactly to a Ring-LWE sample associated to $\boldsymbol{s}$ and q ($\boldsymbol{a} \xleftarrow{\$} \mathcal{U}_q$ and $\boldsymbol{e} \xleftarrow{\$} \chi_{err}$). The relinearization key $\overrightarrow{\texttt{rlk}} = ([\mathcal{P}_{\omega,q}(\boldsymbol{s}^2) + \overrightarrow{\boldsymbol{a}}\cdot\boldsymbol{s} + \overrightarrow{\boldsymbol{e}}]_q, -\overrightarrow{\boldsymbol{a}}) \in \mathcal{R}_q^{\ell_{\omega,q}} \times \mathcal{R}_q^{\ell_{\omega,q}}$ is modified in the same way as the public key ($\overrightarrow{\boldsymbol{a}} \in \mathcal{R}_q^{\ell_{\omega,q}}$ sampled from $\mathcal{U}_q^{\ell_{\omega,q}}$ and $\overrightarrow{\boldsymbol{e}} \in \mathcal{R}_q^{\ell_{\omega,q}}$ sampled from $\chi_{err}^{\ell_{\omega,q}}$).

Algorithm 8 Key Generation FV

Require: the public parameters $\texttt{params}_{\texttt{FV}}$.
Ensure: secret key $\texttt{sk}$, public key $\texttt{pk}$ and relinearization key $\overrightarrow{\texttt{rlk}}$
function KeyGen$_{\texttt{FV}}$($\texttt{params}_{\texttt{FV}}$)
$\boldsymbol{s} \xleftarrow{\$} \chi_{key}$
$\texttt{sk} \leftarrow \boldsymbol{s}$
$(\boldsymbol{a}, \boldsymbol{e}) \xleftarrow{\$} \mathcal{U}_q \times \chi_{err}$
$\texttt{pk} \leftarrow ([\boldsymbol{a}\cdot\boldsymbol{s}+\boldsymbol{e}]_q, -\boldsymbol{a})$
$(\overrightarrow{\boldsymbol{a}}, \overrightarrow{\boldsymbol{e}}) \xleftarrow{\$} \mathcal{U}_q^{\ell_{\omega,q}} \times \chi_{err}^{\ell_{\omega,q}}$
$\overrightarrow{\texttt{rlk}} \leftarrow \left(\left[\mathcal{P}_{\omega,q}(\boldsymbol{s}^2) + \overrightarrow{\boldsymbol{a}}\cdot\boldsymbol{s} + \overrightarrow{\boldsymbol{e}}\right]_q, -\overrightarrow{\boldsymbol{a}}\right)$
return $(\texttt{sk}, \texttt{pk}, \overrightarrow{\texttt{rlk}})$

Encryption A message $\boldsymbol{m} \in \mathcal{R}_t$ is encrypted under the public key $\texttt{pk}$ in a ciphertext $\texttt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$ as follows:

$$\texttt{ct} = \left([\Delta[\boldsymbol{m}]_t + \boldsymbol{u}\cdot\mathbf{pk}_0 + \boldsymbol{e}_0]_q, [\boldsymbol{u}\cdot\mathbf{pk}_1 + \boldsymbol{e}_1]_q\right)$$

with $\Delta = \lfloor q/t \rfloor$, $\boldsymbol{u} \xleftarrow{\$} \chi_{key}$ and $\boldsymbol{e}_0, \boldsymbol{e}_1 \xleftarrow{\$} \chi_{err}$. Like for BGV, the noise contained in a ciphertext $\texttt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ is revealed once we evaluate $\texttt{ct}$ on the secret key $\boldsymbol{s}$:

$$\begin{aligned} \boldsymbol{c}_0 + \boldsymbol{c}_1\cdot\boldsymbol{s} &\equiv \Delta[\boldsymbol{m}]_t + \boldsymbol{u}\cdot\mathbf{pk}_0 + \boldsymbol{e}_0 + (\boldsymbol{u}\cdot\mathbf{pk}_1 + \boldsymbol{e}_1)\cdot\boldsymbol{s} \bmod q \\ &\equiv \Delta[\boldsymbol{m}]_t + \boldsymbol{v} \bmod q \end{aligned} \tag{1.2.11}$$

with $\boldsymbol{v} = \boldsymbol{u}\cdot\boldsymbol{e} + \boldsymbol{e}_1\cdot\boldsymbol{s} + \boldsymbol{e}_0$ the noise of a ciphertext "freshly" encrypted which has its norm bounded by:

$$\|\boldsymbol{v}\|_\infty \leq B_{err}(2\delta_{\mathcal{R}}B_{key} + 1) \tag{1.2.12}$$

Decryption Once again evaluating a ciphertext $\texttt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$ on the secret key almost reveals the message (cf. (1.2.11)). However this time one cannot simply retrieve the message by taking the result modulo t. We first need to get rid of the factor Δ by scaling the result by $t/q \simeq \Delta^{-1}$ and rounding the result:

Algorithm 9 Encryption FV

Require: message $\boldsymbol{m} \in \mathcal{R}_t$, public key $\mathtt{pk} = (\mathbf{pk}_0, \mathbf{pk}_1)$ and the public parameters $\mathtt{params}_{\mathtt{FV}}$.
Ensure: ciphertext $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ encrypting the message $\boldsymbol{m}$
function $\textsc{Enc}_{\mathtt{FV}}(\boldsymbol{m}, \mathtt{pk}, \mathtt{params}_{\mathtt{FV}})$
$\boldsymbol{u} \xleftarrow{\$} \chi_{key}$
$(\boldsymbol{e}_0,\ \boldsymbol{e}_1) \xleftarrow{\$} \chi_{err}^2$
return $\mathtt{ct} = \left([\Delta[\boldsymbol{m}]_t + \boldsymbol{u} \cdot \mathbf{pk}_0 + \boldsymbol{e}_0]_q, [\boldsymbol{u} \cdot \mathbf{pk}_1 + \boldsymbol{e}_1]_q\right)$

$$\begin{aligned}\left\lfloor \frac{t}{q} [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q \right\rceil &= \left\lfloor \frac{t}{q} (\Delta[\boldsymbol{m}]_t + \boldsymbol{v} + q\boldsymbol{u}) \right\rceil \text{ for some } \boldsymbol{u} \in \mathcal{R} \\ &= \left\lfloor \frac{q - |q|_t}{q} [\boldsymbol{m}]_t + \frac{t}{q}\boldsymbol{v} \right\rceil + t\boldsymbol{u} \\ &= [\boldsymbol{m}]_t + \left\lfloor \frac{t\boldsymbol{v} - |q|_t [\boldsymbol{m}]_t}{q} \right\rceil + t\boldsymbol{u}\end{aligned}$$

the last rounding will vanish if:

$$\left\| \frac{t\boldsymbol{v} - |q|_t [\boldsymbol{m}]_t}{q} \right\|_\infty < \frac{1}{2} \ \Leftrightarrow\ \left\| \boldsymbol{v} - \frac{|q|_t}{t} [\boldsymbol{m}]_t \right\|_\infty < \frac{q}{2t}$$

which is ensured as long as:

$$\|\boldsymbol{v}\|_\infty < \frac{q}{2t} - \frac{|q|_t}{2} \ = B_{dec} \ \left(\simeq \frac{\Delta}{2}\right) \tag{1.2.13}$$

Finally, if inequality (1.2.13) holds, one just has to take the result of the rounding modulo t to get back the message:

$$\left[\left\lfloor \frac{t}{q} [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q \right\rceil \right]_t = [[\boldsymbol{m}]_t + t\boldsymbol{u}]_t = [\boldsymbol{m}]_t$$

As a consequence, to ensure correct decryption of fresh ciphertexts one needs to have:

$$B_{err}(2\delta_{\mathcal{R}} B_{key} + 1) < B_{dec} \tag{1.2.14}$$

Algorithm 10 Decryption FV

Require: ciphertext $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$, secret key $\mathtt{sk} = \boldsymbol{s}$ and the public parameters $\mathtt{params}_{\mathtt{FV}}$.
Ensure: message $[\boldsymbol{m}]_t$ (if equation (1.2.13) is satisfied).
function $\textsc{Dec}_{\mathtt{FV}}(\mathtt{ct}, \mathtt{sk}, \mathtt{params}_{\mathtt{FV}})$
$\boldsymbol{m}' \leftarrow \left\lfloor \frac{t}{q} \cdot [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q \right\rceil$
return $[\boldsymbol{m}']_t$

Addition The addition of ciphertexts $\mathtt{ct}$ and $\mathtt{ct}'$ encrypting $\boldsymbol{m}$ and $\boldsymbol{m}'$ respectively is done exactly like in BGV:

$$\begin{aligned}\boldsymbol{c}_0 + \boldsymbol{c}_0' + (\boldsymbol{c}_1 + \boldsymbol{c}_1') \cdot \boldsymbol{s} &\equiv \Delta\left([\boldsymbol{m}]_t + [\boldsymbol{m}']_t\right) + \boldsymbol{v} + \boldsymbol{v}' \bmod q \\ &\equiv \Delta[\boldsymbol{m} + \boldsymbol{m}']_t + \boldsymbol{v} + \boldsymbol{v}'(q - |q|_t)\boldsymbol{u} \bmod q \\ &\equiv \Delta[\boldsymbol{m} + \boldsymbol{m}']_t + \boldsymbol{v} + \boldsymbol{v}' - |q|_t\boldsymbol{u} \bmod q\end{aligned}$$

where $[\boldsymbol{m}]_t + [\boldsymbol{m}']_t = [\boldsymbol{m} \cdot \boldsymbol{m}']_t + \boldsymbol{u}$ with $\|\boldsymbol{u}\|_\infty \leq 1$, thus:

$$\mathtt{ct}_{\text{add}} = ([\boldsymbol{c}_0 + \boldsymbol{c}_0']_q, [[\boldsymbol{c}_1 + \boldsymbol{c}_1']_q])$$

is an encryption of $\boldsymbol{m} + \boldsymbol{m}'$ and its noise is roughly the sum of the noises of $\mathtt{ct}$ and $\mathtt{ct}'$:

$$\boldsymbol{v}_{\text{add}} = \boldsymbol{v} + \boldsymbol{v}' + t\boldsymbol{u}$$

of norm bounded by:

$$\|\boldsymbol{v}_{\text{add}}\|_\infty \leq \|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}\|_\infty + t \tag{1.2.15}$$

Algorithm 11 Addition FV

Require: ciphertexts $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ and $\mathtt{ct}' = (\boldsymbol{c}_0', \boldsymbol{c}_1')$ encrypting messages $\boldsymbol{m}$ and $\boldsymbol{m}'$ respectively and the public parameters $\mathtt{params}_{\mathtt{FV}}$.
Ensure: ciphertext $\mathtt{ct}_{\text{add}}$ encrypting $\boldsymbol{m} + \boldsymbol{m}'$.
function $\textsc{Add}_{\mathtt{FV}}(\mathtt{ct}, \mathtt{ct}', \mathtt{params}_{\mathtt{FV}})$
$\mathtt{ct}_{\text{add}} \leftarrow \left([\boldsymbol{c}_0 + \boldsymbol{c}_0']_q, [\boldsymbol{c}_1 + \boldsymbol{c}_1']_q\right)$
return $\mathtt{ct}_{\text{add}}$

Multiplication Multiplying ciphertexts like in BGV would make appear a factor Δ^2 together with the message. Therefore to get back a valid ciphertext the product is scaled by t/q, which roughly remove one factor Δ, and then rounded. Before starting, remark that any valid ciphertext $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$ whose noise has norm smaller than $B_{dec} (< \Delta/2)$ satisfies:

$$\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \Delta[\boldsymbol{m}]_t + \boldsymbol{v} + q\boldsymbol{r} \text{ for some } \boldsymbol{r} \in \mathcal{R}$$

with:

$$\|\boldsymbol{r}\|_\infty = \left\| \frac{\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} - \Delta[m]_t - \boldsymbol{v}}{q} \right\|_\infty \leq \underbrace{\frac{\delta_{\mathcal{R}} B_{key} + 1}{2} + 1}_{\|\boldsymbol{r}_d\|_\infty} \tag{1.2.16}$$

Thus: $(\mathtt{ct} \cdot \mathtt{ct}')(\boldsymbol{s}) = (\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}) \cdot (\boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s})$

$$\begin{aligned}
&= (\Delta[\boldsymbol{m}]_t + \boldsymbol{v} + q\boldsymbol{r}) \cdot (\Delta[\boldsymbol{m}']_t + \boldsymbol{v}' + q\boldsymbol{r}') \\
&= \Delta^2[\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t + \Delta\left([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot [\boldsymbol{m}']_t\right) + q\left(\boldsymbol{v} \cdot \boldsymbol{r}' + \boldsymbol{v}' \cdot \boldsymbol{r}\right) + \boldsymbol{v} \cdot \boldsymbol{v}' \\
&\quad + q\Delta\left([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r}\right) + q^2 \boldsymbol{r} \cdot \boldsymbol{r}' \\
&= \Delta^2\left([\boldsymbol{m} \cdot \boldsymbol{m}']_t + t\boldsymbol{r}_m\right) + \Delta\left([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot [\boldsymbol{m}']_t\right) + q\left(\boldsymbol{v} \cdot \boldsymbol{r}' + \boldsymbol{v}' \cdot \boldsymbol{r}\right) + \boldsymbol{v} \cdot \boldsymbol{v}' \\
&\quad + q\Delta\left([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r}\right) + q^2 \boldsymbol{r} \cdot \boldsymbol{r}'
\end{aligned}$$

with $\boldsymbol{r}_m$ like in (1.2.6). Then scaling by t/q and using $t\Delta = q - |q|_t$ leads to:

$$\begin{aligned}
\frac{t}{q}(\mathtt{ct} \cdot \mathtt{ct}')(\boldsymbol{s}) = \Delta[\boldsymbol{m} \cdot \boldsymbol{m}']_t + \left([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot [\boldsymbol{m}']_t\right) + \frac{t}{q}\boldsymbol{v} \cdot \boldsymbol{v}' + t\left(\boldsymbol{v} \cdot \boldsymbol{r}' + \boldsymbol{v}' \cdot \boldsymbol{r}\right) + tq\boldsymbol{r} \cdot \boldsymbol{r}' \\
+ \Delta t\left([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r} + \boldsymbol{r}_m\right) - \frac{|q|_t}{q}\left(\Delta[\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t + [\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot [\boldsymbol{m}']_t\right)
\end{aligned}$$

Assuming the ciphertexts in input could decrypt correctly, i.e. have noise $\boldsymbol{v}$, $\boldsymbol{v}$' smaller in norm than B_{dec} and thus $\Delta/2$, we can write $\boldsymbol{v} \cdot \boldsymbol{v}' = [\boldsymbol{v}]_\Delta \cdot [\boldsymbol{v}']_\Delta = [\boldsymbol{v} \cdot \boldsymbol{v}']_\Delta + \Delta\boldsymbol{r}_v$ with:

$$\|\boldsymbol{r}_v\|_\infty \leq \frac{\delta_{\mathcal{R}}\|\boldsymbol{v}\|_\infty\|\boldsymbol{v}'\|_\infty}{\Delta} + \frac{1}{2} < \frac{\delta_{\mathcal{R}} \min\{\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}'\|_\infty\}}{2} + \frac{1}{2}$$

$$\implies 2\|\boldsymbol{r}_v\|_\infty < \delta_{\mathcal{R}} \min\{\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}'\|_\infty\} + 1 \implies 2\|\boldsymbol{r}_v\|_\infty \leq \delta_{\mathcal{R}} \min\{\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}\|'_\infty\} \tag{1.2.17}$$

and we obtain:

$$\begin{aligned}
\frac{t}{q}(\mathtt{ct} \cdot \mathtt{ct}')(\boldsymbol{s}) = \Delta[\boldsymbol{m} \cdot \boldsymbol{m}']_t + \left([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot [\boldsymbol{m}']_t\right) + t\left(\boldsymbol{v} \cdot \boldsymbol{r}' + \boldsymbol{v}' \cdot \boldsymbol{r}\right) \\
+ (q - |q|_t)\left([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r} + \boldsymbol{r}_m\right) + \boldsymbol{r}_v + tq\boldsymbol{r} \cdot \boldsymbol{r}' \\
+ \underbrace{\frac{t}{q}[\boldsymbol{v} \cdot \boldsymbol{v}']_\Delta - \frac{|q|_t}{q}\left(\Delta[\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t + [\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot [\boldsymbol{m}']_t + \boldsymbol{r}_v\right)}_{\boldsymbol{r}_r}
\end{aligned}$$

with:

$$\begin{aligned}
\|\boldsymbol{r}_r\|_\infty &< \frac{1}{q}\left(\frac{\Delta t}{2} + |q|_t\left(\frac{\delta_{\mathcal{R}}\Delta t^2}{4} + \frac{\delta_{\mathcal{R}} t}{2}(\|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}'\|_\infty) + \|\boldsymbol{r}_v\|_\infty\right)\right) \\
&< \frac{1}{2} + |q|_t\delta_{\mathcal{R}}\left(\frac{t}{4} + \frac{1}{2} + \frac{1}{4t}\right) < \frac{1}{2} + |q|_t\delta_{\mathcal{R}} t
\end{aligned}$$

hence:

$$\|\boldsymbol{r}_r\|_\infty \leq |q|_t\delta_{\mathcal{R}} t \tag{1.2.18}$$

The rounding procedure introduces an error $\boldsymbol{r}_a \in \mathbb{Q}[X]/(\boldsymbol{\Phi}_m(X))$ such that:

$$\frac{t}{q}(\mathtt{ct} \cdot \mathtt{ct}')(\boldsymbol{s}) = \left\lfloor \frac{t}{q}\boldsymbol{c}_0 \cdot \boldsymbol{c}'_0 \right\rceil + \left\lfloor \frac{t}{q}(\boldsymbol{c}_0 \cdot \boldsymbol{c}'_1 + \boldsymbol{c}_1 \cdot \boldsymbol{c}'_0) \right\rceil \cdot \boldsymbol{s} + \left\lfloor \frac{t}{q}\boldsymbol{c}_1 \cdot \boldsymbol{c}'_1 \right\rceil \cdot \boldsymbol{s}^2 + \boldsymbol{r}_a$$

which has its norm bounded by:

$$\|\boldsymbol{r}_a\|_\infty \leq \frac{1 + \delta_{\mathcal{R}} B_{key} + \delta_{\mathcal{R}}^2 B_{key}^2}{2} \tag{1.2.19}$$

Now, by rounding the above expression and by considering its remainder modulo q we get:

$$\begin{aligned}
&\left\lfloor \frac{t}{q}\boldsymbol{c}_0 \cdot \boldsymbol{c}_0' \right\rceil + \left\lfloor \frac{t}{q}(\boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0') \right\rceil \cdot \boldsymbol{s} + \left\lfloor \frac{t}{q}\boldsymbol{c}_1 \cdot \boldsymbol{c}_1' \right\rceil \cdot \boldsymbol{s}^2 = \frac{t}{q}(\mathtt{ct} \cdot \mathtt{ct}')(\boldsymbol{s}) - \boldsymbol{r}_a \bmod q \\
&= \Delta[\boldsymbol{m} \cdot \boldsymbol{m}']_t + ([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot [\boldsymbol{m}']_t) + t\,(\boldsymbol{v} \cdot \boldsymbol{r}' + \boldsymbol{v}' \cdot \boldsymbol{r}) - |q|_t\,([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r} + \boldsymbol{r}_m) \\
&\quad + \boldsymbol{r}_v + \boldsymbol{r}_r - \boldsymbol{r}_a \bmod q
\end{aligned}$$

So finally we obtain:

$$\left\lfloor \frac{t}{q}\boldsymbol{c}_0 \cdot \boldsymbol{c}_0' \right\rceil + \left\lfloor \frac{t}{q}(\boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0') \right\rceil \cdot \boldsymbol{s} + \left\lfloor \frac{t}{q}\boldsymbol{c}_1 \cdot \boldsymbol{c}_1' \right\rceil \cdot \boldsymbol{s}^2 = \Delta[\boldsymbol{m} \cdot \boldsymbol{m}']_t + \boldsymbol{v}_{\text{mult}} \bmod q$$

which means that:

$$\hat{\mathtt{ct}} = \left(\left[\left\lfloor \frac{t}{q}\boldsymbol{c}_0 \cdot \boldsymbol{c}_0' \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q}(\boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0') \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q}\boldsymbol{c}_1 \cdot \boldsymbol{c}_1' \right\rceil \right]_q \right) \tag{1.2.20}$$

is a degree 2 ciphertext which encrypts $\boldsymbol{m} \cdot \boldsymbol{m}'$ and whose inherent noise is:

$$\boldsymbol{v}_{\text{mult}} = ([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot [\boldsymbol{m}']_t) + t\,(\boldsymbol{v} \cdot \boldsymbol{r}' + \boldsymbol{v}' \cdot \boldsymbol{r}) - |q|_t\,([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r} + \boldsymbol{r}_m) + \boldsymbol{r}_v + \boldsymbol{r}_r - \boldsymbol{r}_a$$

by using the bounds (1.2.6), (1.2.16), (1.2.17), (1.2.18) and (1.2.19) we can show:

$$\begin{aligned}
\|\boldsymbol{v}_{\text{mult}}\|_\infty \leq\ & \delta_{\mathcal{R}} t \left(\frac{\delta_{\mathcal{R}} B_{key}}{2} + \frac{3}{2} \right) (\|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}'\|_\infty) + \frac{\delta_{\mathcal{R}} \min\{\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}'\|_\infty\}}{2} \\
&+ |q|_t \delta_{\mathcal{R}} t \left(\frac{\delta_{\mathcal{R}} B_{key}}{2} + 2 \right) + \frac{\delta_{\mathcal{R}}^2 B_{key}^2 + \delta_{\mathcal{R}} B_{key} + 1}{2}
\end{aligned} \tag{1.2.21}$$

Algorithm 12 Multiplication FV

Require: ciphertexts $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ and $\mathtt{ct}' = (\boldsymbol{c}_0', \boldsymbol{c}_1')$ encrypting messages $\boldsymbol{m}$ and $\boldsymbol{m}'$ respectively and the public parameters $\mathtt{params}_{\mathtt{FV}}$.
Ensure: degree 2 ciphertext $\mathtt{ct}_{\text{mult}}$ encrypting $\boldsymbol{m} \cdot \boldsymbol{m}'$.
function $\textsc{Mult}_{\mathtt{FV}}(\mathtt{ct}, \mathtt{ct}', \mathtt{params}_{\mathtt{FV}})$
$\tilde{\mathtt{ct}} \leftarrow (\boldsymbol{c}_0 \cdot \boldsymbol{c}_0',\ \boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0',\ \boldsymbol{c}_1 \cdot \boldsymbol{c}_1')$
$\mathtt{ct}_{\text{mult}} \leftarrow \left(\left[\left\lfloor \frac{t}{q}\tilde{\mathtt{ct}}_0 \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q}\tilde{\mathtt{ct}}_1 \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q}\tilde{\mathtt{ct}}_2 \right\rceil \right]_q \right)$
return $\mathtt{ct}_{\text{mult}}$

Relinearization The relinearization step of FV is introduced for the same reasons than in BGV and works exactly the same. By computing:

$$\tilde{\mathtt{ct}} = \left(\left[\boldsymbol{c}_0 + \left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_0} \right\rangle \right]_q, \left[\boldsymbol{c}_1 + \left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_1} \right\rangle \right]_q \right) \in \mathcal{R}_q^2$$

we obtain a degree 1 ciphertext, which encrypts the same message than ct but with an additional term $\boldsymbol{b}_{\text{relin}}$ in the noise:

$$\tilde{\boldsymbol{v}} = \boldsymbol{v} + \boldsymbol{b}_{\text{relin}}$$

whose norm is smaller than:

$$\|\boldsymbol{b}_{\text{relin}}\|_\infty \leq \frac{\delta_{\mathcal{R}} \ell_{\omega,q_i} \omega B_{err}}{2}$$

Algorithm 13 Relinearization FV

Require: degree 2 ciphertext $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2)$, the relinearization key $\overrightarrow{\mathtt{rlk}} = (\overrightarrow{\mathtt{rlk}_0}, \overrightarrow{\mathtt{rlk}_1})$ and the public parameters $\mathtt{params}_{\mathtt{FV}}$.
Ensure: degree 1 ciphertext $\mathtt{ct}' = (\boldsymbol{c}_0', \boldsymbol{c}_1')$ encrypting the same message than ct.
function RELINEARIZATION$_{\mathtt{FV}}$($\mathtt{ct}, \overrightarrow{\mathtt{rlk}}, \mathtt{params}_{\mathtt{FV}}$)
$\mathtt{ct}' \leftarrow \left(\left[\boldsymbol{c}_0 + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_0} \right\rangle \right]_q, \left[\boldsymbol{c}_1 + \left\langle \mathcal{D}_{\omega,q_i}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_1} \right\rangle \right]_q \right)$
return ct′

Multiplicative depth Contrary to BGV which is built such that the multiplicative depth is the length of the moduli chain minus one, the multiplicative depth in FV can not be read directly from the parameters. However we can compute it by using the same method than [BLLN13]. Let us assume that both ciphertexts, in input of the multiplication procedure, have a noise smaller than V in norm. In the case of fresh ciphertext V is smaller than $V_{init} = B_{err}(2\delta_{\mathcal{R}} B_{key} + 1)$. From the bounds on the noise given above we can deduce that the size of the noise after one multiplication and relinearization is bounded by $C_1 V + C_2$ with:

$$\begin{cases} C_1 = \delta_{\mathcal{R}} t \left(\delta_{\mathcal{R}} B_{key} + 3 \right) + \frac{\delta_{\mathcal{R}}}{2} \\ C_2 = |q|_t \delta_{\mathcal{R}} t \left(\frac{\delta_{\mathcal{R}} B_{key}}{2} + 2 \right) + \frac{\delta_{\mathcal{R}}^2 B_{key}^2 + \delta_{\mathcal{R}} B_{key} + 1}{2} + \frac{\delta_{\mathcal{R}} \ell_{\omega,q_i} \omega B_{err}}{2} \end{cases}$$

Therefore after L multiplications the noise is bounded by:

$$C_1^L V + C_2 \sum_{i=0}^{L-1} C_1^i = C_1^L V + C_2 \frac{C_1^L - 1}{C_1 - 1}$$

Thus the theoretical multiplicative depth corresponds to the biggest integer $L_{\max}$ such that:

$$C_1^{L_{\max}} V_{init} + C_2 \frac{C_1^{L_{\max}} - 1}{C_1 - 1} < B_{dec} \tag{1.2.22}$$

1.3 Classical arithmetic in cyclotomic rings

In order to use in practice Ring-LWE based cryptography, one needs to implement basic operations for the arithmetic on cyclotomic rings whose elements can be seen as polynomials of degree smaller than n with coefficients taken modulo q. However to be able to tolerate large noise growth, q is usually chosen large (several hundreds of bits). Besides the size of q, the rank of the associated lattice, which corresponds to n, has to be high enough to meet the security requirements (usually between 2^{10} and 2^{15}). This section aims to present the different tools which are used in practice to get an efficient arithmetic on the coefficients and on the polynomials with such parameters. We end our exposition by showing how to encode several messages into a single one while processing them independently through a single ciphertext. All these tools rely on the Chinese Remainder Theorem (CRT) that we recall below for the sake of completeness. We state it in the case of commutative rings which have a neutral element for the multiplication $1 \neq 0$.

Definition 1.3.1. Let R be a commutative ring, two ideals I and J of R are said to be *coprime* if there exists $x \in I$ and $y \in J$ such that $x + y = 1$.

Definition 1.3.2. Let R be a commutative ring and I and J two ideals of R, the *product* $I \cdot J$ of the ideals I and J is the ideal of R formed by the set of all the finite sums of elements of the form xy for $x \in I$ and $y \in J$.

Theorem 1.3.3 ([DF04] section 7.6)**.** Let $I_1, \ldots, I_k$ be k ideals of R that are pairwise coprime and let $I = I_1 \cdot I_2 \cdots I_k$ be their product, then we have the following ring isomorphism:

$$CRT : \left| \begin{array}{ccc} R/I & \xrightarrow{\cong} & R/I_1 \times \cdots \times R/I_k \\ x \bmod I & \mapsto & (x \bmod I_1, \ \ldots \ , x \bmod I_k) \end{array} \right.$$

Remark 1.3.4. This ring isomorphism allows to carry the ring operations on the residues modulo each "small" ideal I_i independently and thus in parallel.

1.3.1 On coefficients: Residue Number System

In order to reach an efficient arithmetic in a polynomial ring, the first thing to do is to get an efficient one in the ring of the coefficients. In our case the coefficients belong to $\mathbb{Z}/q\mathbb{Z}$ with q an integer of possibly several hundreds of bits without any restriction on its shape (Remark 1.1.38). Asymptotic complexities of operations modulo an integer q for naive algorithms are given in Table 1, for further details on these algorithms one can refer to [MVO96].

As illustrated in Table 1, the bottleneck of modular arithmetic is multiplication (and inversion) whose time complexity is quadratic in the size of q. Therefore for evaluating the cost of an algorithm we usually only consider the number of multiplications/inversions and do not take into account the additions/subtraction's whose cost is negligible compared to multiplications.

Operation		Complexity
Modular addition	$a + b \bmod q$	$\mathcal{O}(\log_2 q)$
Modular subtraction	$a - b \bmod q$	$\mathcal{O}(\log_2 q)$
Modular multiplication	$ab \bmod q$	$\mathcal{O}((\log_2 q)^2)$
Modular inversion (when possible)	$a^{-1} \bmod q$	$\mathcal{O}((\log_2 q)^2)$

Table 1: Asymptotic complexity of basic operations in $\mathbb{Z}/q\mathbb{Z}$

Although dividing the size of q by a constant k does not affect the asymptotic complexity of the operations, it brings an important gain in practice. Hence if some application requires a large modulus q, one has an incentive to decompose it in a product of k smaller prime moduli and use the representation given via the CRT a.k.a Residue Number System (RNS) ([Gar59]). Unless stated otherwise, from now we assume $q = q_1 \dots q_k$ to be a product of k prime moduli of same size. Hence the polynomials of $\mathcal{R}_q$ are represented through k polynomials of same degree but with smaller coefficients thanks to the ring isomorphism given by the CRT:

$$RNS_{q=q_1 \dots q_k} : \left| \begin{array}{ccc} \mathcal{R}_q & \xrightarrow{\cong} & \mathcal{R}_{q_1} \times \cdots \times \mathcal{R}_{q_k} \\ a & \mapsto & (a \bmod q_1, \ \dots \ , a \bmod q_k) \end{array} \right. \tag{1.3.1}$$

Remark 1.3.5. The attentive reader should have noticed that to get the isomorphism the moduli q_i only need to be pairwise coprime and not prime. However, for reasons that will be explained in the next section they are chosen prime.

RNS representation turns out to be very efficient to compute sums and products to the point where it is commonly used in implementation of RSA ([BI04]) or for elliptic curve cryptography ([BDEM06]). Furthermore it offers a competitive alternative for issues concerning material implementations with low power consumption [FP97]. However despite its computational efficiency the RNS representation suffers from the drawbacks of non-positional representation systems which limit its practicality. Indeed unlike positional representation such as the binary representation, non-positional representations do not allow to perform any comparison between numbers. Therefore operations like non-exact divisions or roundings require to invert the isomorphism (1.3.1) or at least to convert the residues to an other positional representation like Mix Radix System ([ST67]).

Definition 1.3.6. An *(RNS) basis* is a finite set of integers $\mathcal{B} = \{b_1, \dots, b_l\}$ pairwise coprime and $B = b_1 \cdots b_l$ is called the *(RNS) modulus* of the basis. The set of residues of an element $a \in \mathcal{R}_q$ in basis $\mathcal{B}$ is denoted $a_{\mathcal{B}}$.

In the general case, RNS can be used to represent integers and not necessary modular integers. This only requires to use a basis large enough to represent our elements, i.e. such that the modulus of the basis is larger than the biggest element we may need to represent. This condition is needed to avoid an eventual overflow of the basis capacity which would result in a reduction by the modulus of the basis. However since in our case we only want to represent integers modulo q, we do not have to fear an eventual reduction modulo q and

we can use $\{q_1, \dots, q_k\}$ as RNS basis. Throughout the rest of this thesis depending on the context, the letter q may refer either to the product $q_1 \cdots q_k$ or the RNS basis $\{q_1, \dots, q_k\}$.

Proposition 1.3.7. Let $\boldsymbol{a} \in \mathcal{R}_q$ be represented by its residues $\boldsymbol{a}_q$ in base q, then the polynomial of $\mathcal{R}$ defined by:

$$\texttt{PartialInv}(\boldsymbol{a}_q) = \sum_{i=1}^{k} \left| |\boldsymbol{a}|_{q_i} \frac{q_i}{q} \right|_{q_i} \frac{q}{q_i} \tag{1.3.2}$$

is congruent to $\boldsymbol{a}$ modulo q and has its coefficients belonging to $[0, kq) \cap \mathbb{Z}$.

Proof. One can verify that $|\texttt{PartialInv}(\boldsymbol{a}_q)|_{q_i} = |\boldsymbol{a}|_{q_i}$ for every modulus q_i thus it is congruent to $\boldsymbol{a}$ modulo q. Its coefficients are positive by definition and by applying the triangular inequality to upper-bound them we obtain the result. □

Remark 1.3.8. The values $|q_i/q|_{q_i}$ can be precomputed for every q_i so that the partial inversion requires k multiplications for each coefficients thus kn multiplications in total.

By computing `PartialInv` one does not necessary obtain the polynomial $\boldsymbol{a} \in \mathcal{R}_q$ but one of its representatives whose coefficients are not reduced modulo q. In order to get back $\boldsymbol{a}$, the result has to be reduced modulo q:

$$\boldsymbol{a} = \texttt{PartialInv}(\boldsymbol{a}_q) - q \cdot \boldsymbol{\alpha}(\boldsymbol{a}_q) \tag{1.3.3}$$

with $\boldsymbol{\alpha}(\boldsymbol{a}_q) = \left\lfloor \dfrac{\texttt{PartialInv}(\boldsymbol{a}_q)}{q} \right\rfloor$ which has its coefficients in $[0, k) \cap \mathbb{Z}$ (eq. 1.3.2).

As illustrated in Chapter 2, some computations in RNS require to change the RNS basis which means to convert the residues of $\boldsymbol{a} \in \mathcal{R}_q$, given in basis q, to an other basis $\mathcal{B} = \{b_1, \dots, b_l\}$ which can be achieved by computing:

$$\texttt{FastBconv}(\boldsymbol{a}, q, \mathcal{B}) = \left(\texttt{PartialInv}(\boldsymbol{a}_q) \bmod b_i\right)_{b \in \mathcal{B}}. \tag{1.3.4}$$

Even though this computation is quiet efficient (kln modular multiplications) it does not allow to obtain exactly the residues of $|\boldsymbol{a}|_q$ in base $\mathcal{B}$ but those of $|\boldsymbol{a}|_q + \alpha(\boldsymbol{a}_q)q$ with $\boldsymbol{\alpha}(\boldsymbol{a}_q) \in [0, k)$ (c.f. eq. (1.3.3)). Getting an exact conversion would require to compute $\boldsymbol{\alpha}(\boldsymbol{a}_q)$ with fix or floating point arithmetic at first glance and thus cannot be achieved directly in RNS. In [SK89], Shenoy and Kumaresan proposed a method to retrieve $\boldsymbol{\alpha}(\boldsymbol{a}_q)$ directly in RNS by introducing an extra modulus in the initial basis. More precisely, let $\mathcal{B} = \{b_1, \dots, b_k\}$ and $\mathcal{B}'$ be two RNS bases used to represent elements of $\mathcal{R}$ and $b_{\texttt{sk}}$ a modulus coprime to the product $B = b_1 \cdots b_k$. Let us assume that we want to convert exactly the residues of $\boldsymbol{a} \in \mathcal{R}$ from basis $\mathcal{B}$ to the basis $\mathcal{B}'$, if we know the residue of $\boldsymbol{a}$ modulo $b_{\texttt{sk}}$ before the conversion, i.e. if we have the residues of $\boldsymbol{a}$ in $\mathcal{B}_{\texttt{sk}} = \mathcal{B} \cup \{b_{\texttt{sk}}\}$ then we can compute $|\boldsymbol{\alpha}(\boldsymbol{a}_{\mathcal{B}})|_{b_{\texttt{sk}}}$ by inverting equation (1.3.3):

$$|\boldsymbol{\alpha}(\boldsymbol{a}_{\mathcal{B}})|_{b_{\texttt{sk}}} = \left| B^{-1} (\texttt{PartialInv}(\boldsymbol{a}_{\mathcal{B}}) - |\boldsymbol{a}|_{b_{\texttt{sk}}}) \right|_{b_{\texttt{sk}}} \tag{1.3.5}$$

Since $\boldsymbol{\alpha}(\boldsymbol{a}_{\mathcal{B}}) \in [0, k) \cap \mathbb{Z}$, if we choose $b_{\mathtt{sk}}$ coprime to B and such that $b_{\mathtt{sk}} \geq k$ then $|\boldsymbol{\alpha}(\boldsymbol{a}_{\mathcal{B}})|_{b_{\mathtt{sk}}} = \boldsymbol{\alpha}(\boldsymbol{a}_{\mathcal{B}})$ which means that we are able to correct the error and perform an exact conversion from $\mathcal{B}$ to $\mathcal{B}'$ at the price of some extra-computations.

1.3.2 On polynomials: Number Theoretic Transform

Multiplication of elements in $\mathcal{R}_q$ corresponds to the multiplication of two polynomials of degree $n-1$ followed by a reduction modulo $\boldsymbol{\Phi}_m$. Since n is large in practice, these operations are the major bottleneck for the efficiency of Ring-LWE based schemes. Several algorithms to perform a polynomial product exist, the number of multiplications asymptotically required by the most important of them is given in Table 2.

Algorithm	Complexity
Naive	$\mathcal{O}(n^2)$
Karatsuba	$\mathcal{O}(n^{\log_2(3)})$ $(\approx n^{1.585})$
Toom-Cook	$\mathcal{O}(n^{log_3(5)})$ $(\approx n^{1.465})$
Fast Fourier Transform (FFT)	$\mathcal{O}(n \log_2(n))$

Table 2: Number of multiplications modulo q asymptotically required for performing a product of two degree n polynomials with classical algorithms

As illustrated in Table 2, FFT is the most efficient asymptotically and thus is usually the one used for Ring-LWE arithmetic. We would like to highlight that the name FFT usually refers to the case where polynomials have real or complex coefficients, but when the coefficients belong to a finite field the algorithm is called Number Theoretic Transform (NTT). The principle of the algorithm is to evaluate the input polynomials on the n-roots of unity, perform the product coefficient-wise and interpolate the result. Since multiplication coefficient-wise requires a linear number of multiplications, the bottleneck of this procedure is the evaluation/interpolation of the polynomials. Evaluation of a polynomial of degree $n-1$ can be done optimally in the general case with n multiplications using Horner's method. Hence an evaluation on n different points, required to interpolate a degree $n-1$ polynomial, would require n^2 multiplications with a naive approach. However by using symmetries of the n-roots of unity, Cooley and Tukey showed that we could evaluate/interpolate a polynomial on the n-roots of unity, with $\mathcal{O}(n \log_2(n))$ multiplications asymptotically ([CT65]). This algorithm, known as *Fast Fourier Transform*, is optimal for the 2^d-roots of unity ($d \geq 0$), therefore when $n-1$ is not a power of two the coefficients of input polynomials are usually padded with zeroes up to the next power of two. Let $\mathcal{N}_2$ be the function defined over $\mathbb{N}$ such that for any $n \in \mathbb{N}$ $\mathcal{N}_2(n)$ is the smallest power of two greater than or equal to n.

$$\mathcal{N}_2 : \left| \begin{array}{ccc} \mathbb{N} & \rightarrow & \mathbb{N} \\ n & \mapsto & 2^{\lceil \log_2(n) \rceil} \end{array} \right. \tag{1.3.6}$$

In our context elements of $\mathcal{R}_q$ have degree $n-1$, thus the product of two elements, before the reduction modulo $\boldsymbol{\Phi}_m(X)$, have degree $2n-2$. As a consequence NTT for multiplications

of elements in $\mathcal{R}_q$ must be of size $N = \mathcal{N}_2(2n)$. The last thing to ensure to be able to use the NTT algorithm is the existence of the N-roots of unity in the ambient space.

Lemma 1.3.9. Let q be a prime number, then $\mathbb{F}_q = \mathbb{Z}/q\mathbb{Z}$ has an element of order $N \geq 1$ if and only if $q = 1 \mod N$.

Proof. First notice that the multiplicative group of $\mathbb{F}_q$ is $\mathbb{F}_q - \{0\}$ which has order $q - 1$ and is cyclic. The direct implication is a consequence of Lagrange's theorem, let ζ be an element of order m in $\mathbb{F}_q^*$ then m divides the order of the group $q - 1$. For the second implication, consider a generator g of $\mathbb{F}_q^\times$, since $N \mid q - 1$ then $\psi = g^{\frac{q-1}{N}}$ has order N. □

Remark 1.3.10. In our case the arithmetic on $\mathcal{R}_q$ is decomposed on k smaller arithmetics over the $\mathcal{R}_{q_i}$ through the RNS representation of the coefficients (c.f. section 1.3.1). Therefore, to be able to use the NTT algorithm on each $\mathcal{R}_{q_i}$ we need to choose all the moduli q_i prime and congruent to 1 modulo N, with $N = \mathcal{N}_2(2n)$.

Throughout the rest of this section q will denote a prime congruent to 1 modulo $N = \mathcal{N}_2(2n)$ and $\psi \in \mathbb{F}_q$ a primitive N-root of unity. The NTT algorithm can be seen as the evaluation of the ring isomorphism given by the CRT:

$$\mathrm{NTT}_{q,N,\psi} : \left| \begin{array}{ccc} \mathbb{F}_q[X]/(X^N - 1) & \xrightarrow{\cong} & \mathbb{F}_q[X]/(X-1) \times \cdots \times \mathbb{F}_q[X]/(X - \psi^{N-1}) \\ \boldsymbol{a} & \mapsto & \left(\boldsymbol{a}(1)\ ,\ \boldsymbol{a}(\psi)\ ,\ \ldots\ ,\ \boldsymbol{a}(\psi^{N-1})\right) \end{array} \right. \tag{1.3.7}$$

Once (1.3.7) is applied we obtain the NTT representation of the polynomials and their product can be performed coordinate-wise thus the time-complexity of the polynomial arithmetic becomes linear in N. Intensive efforts have been made to improve the efficiency of the NTT, either by saving some reductions modulo the prime q [Har14], or by using moduli with special shape [LN16] to allow lazy reductions. When the context is clear, we will denote an NTT transformation of degree N by NTT_N instead of $\mathrm{NTT}_{q,N,\psi}$. Once the inputs are in NTT representation, one can compute the NTT representation of the product $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$ of degree $2n - 2$ through:

$$\mathrm{NTT}_N(\boldsymbol{c}) = \mathrm{NTT}_N(\boldsymbol{a}) \odot \mathrm{NTT}_N(\boldsymbol{b}) \tag{1.3.8}$$

where $\odot$ denotes the component-wise multiplication in $\mathbb{F}_q$. To obtain the value of $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b} \in \mathcal{R}_q$, a second step is needed, which consists of reducing the result of (1.3.8) modulo $\boldsymbol{\Phi}_m(X)$. Further details on this reduction step are given in chapter 3

Before ending this section we would like to draw the reader's attention on the special case occurring when the index m of the cyclotomic is a power of two. In this case $n = m/2$, $N = m$ and $\boldsymbol{\Phi}_m(X) = X^n + 1$ which means that the roots of $\boldsymbol{\Phi}_m$ are the n-roots of -1.

Lemma 1.3.11. Let $m \geq 2$ an even integer, $n = m/2$ and q a prime such that $q = 1 \bmod m$. Let $\psi \in \mathbb{F}_q$ be an element of order m, in particular $\psi^n = -1 \bmod q$, then $\Psi = \psi^2$ is a primitive n-root of unity and $\{\Psi^i \psi\}_{i=0}^{n-1}$ is the set of the n-roots of -1 over $\mathbb{F}_q$.

Proof. ψ is of order m thus ψ^2 is of order $n = m/2$ by definition. It is immediate to show that $(\Psi^i\psi)^n = -1 \bmod q$ for every $i \in [0, n) \cap \mathbb{Z}$. Hence we have n roots of $X^n + 1$ over $\mathbb{F}_q$ which is a field thus we have all of them. □

Thus when m is a power of two and q and $\psi \in \mathbb{F}_q$ are like in Lemma 1.3.11 we can evaluate a polynomial $\boldsymbol{a} \in \mathcal{R}_q$ directly on the roots of $\boldsymbol{\Phi}_m$ by applying an NTT of size $n = N/2$ on $f_\psi(\boldsymbol{a}) = \sum_{i=0}^{n-1} a_i \psi^i X^i \in \mathcal{R}_q$. In this case the product of two polynomials is done directly modulo $X^n + 1 = \boldsymbol{\Phi}_m$ and we can get back the result $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b} \in \mathcal{R}_q$ by just inverting the NTT representation and f_ψ through:

$$\boldsymbol{c} = f_{\psi^{-1}}\left(\mathrm{NTT}^{-1}_{N/2,\psi^2}\left(\mathrm{NTT}_{N/2,\psi^2}(f_\psi(\boldsymbol{a})) \odot \mathrm{NTT}_{N/2,\psi^2}(f_\psi(\boldsymbol{b}))\right)\right) \tag{1.3.9}$$

This technique introduced in [LMPR08], and known as *negative-wrapped convolution*, allows to perform a product in $\mathcal{R}_q$ with NTT of size $n = N/2$ instead of N and without requiring any polynomial reduction. The cheap pre- (resp post-) processing required for the evaluations of f_ψ (resp. $f_\psi^{-1} = f_{\psi^{-1}}$) can be avoided by merging the powers of ψ (resp. ψ^{-1}) with powers of $\Psi = \psi^2$ (resp. Ψ^{-1}) inside the NTT [RVM+14] (resp. inverse NTT [POG15]).

We end this section by summarizing in Figure 3 the different representations used to perform computations in $\mathcal{R}_q$ with $q = q_1 \cdots q_k$ a product of k prime moduli of same size all congruent to 1 modulo $N = \mathcal{N}_2(2n)$.

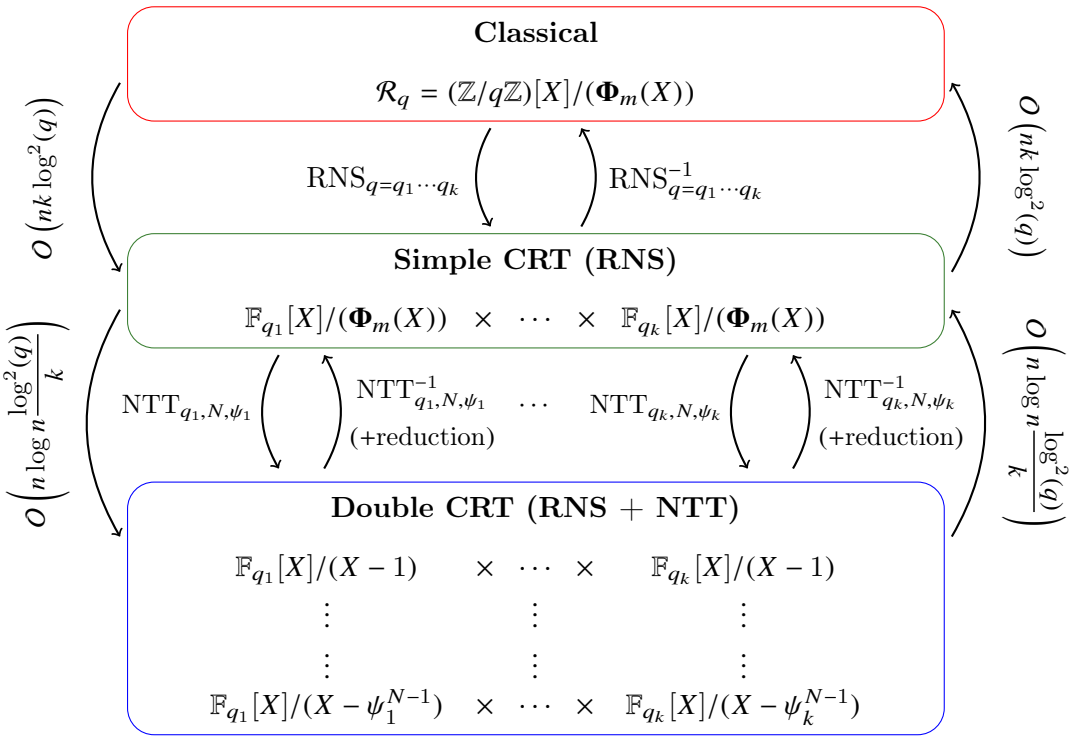


Figure 3: Summary of the different representations used for the arithmetic of $\mathcal{R}_q$

1.3.3 Splitting cyclotomic polynomials for homomorphic encryption

In practice, the efficiency of Ring-LWE based homomorphic encryption schemes is strongly related to the polynomial arithmetic in the ring $\mathcal{R}_q$. Choosing m as a power of 2 allows to use NTTs with the negative wrapped convolution technique and thus to perform one multiplication in $\mathcal{R}_q$ through the operations in (1.3.8) with half-sized NTTs, resulting in a very efficient arithmetic. However, even in this case the efficiency of the schemes remains lowly satisfactory and hardly practical. In [SV14], the authors found a way to amortize the cost of these operations by encoding several messages in a single plaintext element, with a technique called *batching*. The idea is to choose a prime plaintext modulus t such that, $\mathbf{\Phi}_m$ splits modulo t into ℓ distinct irreducible polynomials $\boldsymbol{F}_1$, ...,$\boldsymbol{F}_\ell$, thus a message in the original plaintext space can be split through the CRT into ℓ smaller plaintext spaces.

Proposition 1.3.12. Let t be a prime and m a positive integer not divisible by t. The m-th cyclotomic polynomial $\mathbf{\Phi}_m$ of degree $n = \varphi(m)$ splits modulo t into $\ell = n/d$ irreducible factors of same degree d, where d is the order of t modulo m.

Proof. Let ζ be a root of $\mathbf{\Phi}_m$ in a splitting field K of $\mathbf{\Phi}_m$ over $\mathbb{F}_t$. Since ζ is a root of $\mathbf{\Phi}_m$, it has multiplicative order m in $K^\times$. Let d be the degree of the minimal polynomial of ζ over $\mathbb{F}_t$, then d is the smallest integer such that $\zeta^{t^d} = 1$, i.e. d is the smallest integer such that $m \mid t^d - 1$. □

Hence when the conditions of Proposition 1.3.3 are fulfilled, $\mathbf{\Phi}_m$ splits modulo t in $\boldsymbol{F}_1, \ldots, \boldsymbol{F}_\ell$ and the CRT gives us the ring isomorphism:

$$\mathtt{CRT}_t \left| \begin{array}{ccc} \mathcal{R}_t & \xrightarrow{\cong} & \mathbb{F}_t[X]/(\boldsymbol{F}_1) \times \cdots \times \mathbb{F}_t[X]/(\boldsymbol{F}_\ell) \\ \boldsymbol{a} & \mapsto & (\boldsymbol{a} \bmod \boldsymbol{F}_1, \ldots, \boldsymbol{a} \bmod \boldsymbol{F}_\ell) \end{array} \right. \tag{1.3.10}$$

In this way, ℓ "small" plaintexts $\boldsymbol{m}_1$, ..., $\boldsymbol{m}_\ell$ can be compactly represented as a single polynomial $\boldsymbol{m} \in \mathcal{R}_t$. Afterwards $\boldsymbol{m}$ is encrypted and homomorphic operations applied to this ciphertext operate on each slot individually. This technique is particularly interesting when the messages to encode are small. For instance when evaluating Boolean circuits, i.e. with $t = 2$, the bit to encrypt is encoded on the degree zero coefficient of the message, thus only one coefficient (over n) would be used while with batching one can pack ℓ-bits per plaintext.

Remark 1.3.13. When m is a power of two, since $\mathbf{\Phi}_m(X) = X^{m/2} + 1 \equiv (X+1)^{m/2} \bmod 2$, this technique cannot be used. Thus the efficient arithmetic associated with power-of-two cyclotomics has limited applicability thus for practical application one may considered non-power of two cyclotomics (see for instance: [GHS12b], [HS18], [BJ18], [CGH+18], ...).

With batching, the ℓ slots are processed independently in their ambient space, however for some computations it can be useful to permute the slots directly through the ciphertexts. In [GHS12a], Gentry et al. have shown how to achieve this by using the action of the Galois group of $\mathcal{K}_m = \mathbb{Q}(\zeta_m)$ over $\mathbb{Q}$ on the cyclotomic ring $\mathcal{R} \subsetneq \mathcal{K}_m$. We give an overview of their method in the following.

Let t and m be like in Proposition 1.3.3 and let $\mathcal{G} = \mathrm{Gal}(\mathcal{K}_m/\mathbb{Q})$. We know that $(\mathbb{Z}/m\mathbb{Z})^\times$ is isomorphic to $\mathcal{G}$ through the map $i \mapsto (X \mapsto X^i)$ (theorem 1.1.20). It is noteworthy to notice that elements of the Galois group σ commutes with $\mathtt{CRT}_t$ i.e. for $\boldsymbol{a} \in \mathcal{R}_t$ we have $\sigma(\mathtt{CRT}_t(\boldsymbol{a})) = \mathtt{CRT}_t(\sigma(\boldsymbol{a})) = (\sigma(\boldsymbol{a}_1), \ldots, \sigma(\boldsymbol{a}_\ell))$. Moreover since t generates a subgroup of order d in $(\mathbb{Z}/m\mathbb{Z})^\times$, $\mathcal{G}$ contains a subgroup of order d generated by $X \mapsto X^t$.

Proposition 1.3.14 ([DF04], section 13.5)**.** Let $\mathbb{F}$ be a field of characteristic t, the map:

$$\mathcal{F}_t : \left| \begin{array}{ccc} \mathbb{F} & \to & \mathbb{F} \\ x & \mapsto & x^t \end{array} \right. \tag{1.3.11}$$

is an injective field morphism called the *Frobenius endomorphism* of $\mathbb{F}$, also simply called the *Frobenius* of $\mathbb{F}$. When $\mathbb{F}$ is finite it is an automorphism of $\mathbb{F}$ over its prime field.

Proposition 1.3.15 ([DF04], section 14.1)**.** Let t a prime and $d \geq 1$ an integer. The Galois group of $\mathbb{F}_{t^d}$ over $\mathbb{F}_t$ is cyclic of order d and is generated by the Frobenius endomorphism $\mathcal{F}_t$.

In our case, $X \mapsto X^t$ induces, through $\mathtt{CRT}_t$, the Frobenius automorphism over each field $\mathbb{F}_t[X]/(\boldsymbol{F}_i(X)) \cong \mathbb{F}_{t^d}$ for all the factors $\boldsymbol{F}_i$ of $\boldsymbol{\Phi}_m$ and generates their Galois group over $\mathbb{F}_t$. In particular the group $\mathcal{F} = \langle X \mapsto X^t \rangle$, seen as Galois group, acts transitively on the roots of the factors $\boldsymbol{F}_i$. Thus, by considering the action of the subgroup $\mathcal{F}$ of $\mathcal{G}$ over all the roots of $\boldsymbol{\Phi}_m$, we are able to partition them into ℓ disjoint subset of cardinality d, where each subset corresponds to the roots of a factor $\boldsymbol{F}_i$.

Since $\mathcal{G}$ acts transitively on all the roots of $\boldsymbol{\Phi}_m$, the quotient group $\mathcal{H} = \mathcal{G}/\mathcal{F}$ acts transitively on the set $X_1, \ldots, X_\ell$ where each X_i is a representative of the roots of the factor $\boldsymbol{F}_i$ of $\boldsymbol{\Phi}_m$ modulo t. Since defining a representative X_i essentially means fixing a representation of the field $\mathbb{F}_t[X]/(\boldsymbol{F}_i(X))$, the elements of $\mathcal{H}$ act directly as permutations of the plaintext slots.

Now let us consider ℓ messages $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_\ell$ encoded in $\boldsymbol{m} \in \mathcal{R}_t$ which is encrypted, with the `FV` scheme for instance (can be applied to `BGV` in the same manner), in a ciphertext $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$ with inherent noise $\boldsymbol{v}$ i.e. such that:

$$\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \Delta[\boldsymbol{m}]_t + \boldsymbol{v}$$

where $\boldsymbol{s} \in \mathcal{R}$ is the secret key. For $\sigma_i : X \mapsto X^i \in \mathcal{G}$ and $\boldsymbol{a} \in \mathcal{R}_q$ we denote $\boldsymbol{a}^{(i)} = \sigma_i(\boldsymbol{a})$. Therefore if we apply an element $\sigma_i \in \mathcal{G}$ to this equation we obtain:

$$\boldsymbol{c}_0^{(i)} + \boldsymbol{c}_1^{(i)} \cdot \boldsymbol{s}^{(i)} = \Delta[\boldsymbol{m}^{(i)}]_t + \boldsymbol{v}^{(i)} \bmod q$$

which means that $\mathtt{ct}^{(i)} = (\boldsymbol{c}_0^{(i)}, \boldsymbol{c}_1^{(i)})$ is an encryption of $\boldsymbol{m}^{(i)}$, i.e. $\boldsymbol{m}$ with its slots permuted by σ_i (modulo the action of the Frobenius), and which requires the secret key $\boldsymbol{s}^{(i)}$ to be decrypted. In order to be able to decrypt $\mathtt{ct}^{(i)}$ with the original key $\boldsymbol{s}$ we need to apply a key-switching method whose principle is similar to the relinearization procedure. If the entity which performs the computations on the ciphertexts was given an "encryption" of $\boldsymbol{s}^{(i)}$ as evaluation key:

$$\overrightarrow{\texttt{evk}}^i = \left(\left[\mathcal{P}_{\omega,q}(\boldsymbol{s}^{(i)}) + \vec{\boldsymbol{a}}_i \cdot \boldsymbol{s} + \vec{\boldsymbol{e}}_i\right]_q, -\vec{\boldsymbol{a}}_i\right)$$

with $\vec{\boldsymbol{a}}_i \xleftarrow{\$} \mathcal{U}_q^{\ell_{\omega,q}}$ and $\vec{\boldsymbol{e}}_i \xleftarrow{\$} \chi_{err}^{\ell_{\omega,q}}$ similarly to the relinearization key. Then the ciphertext:

$$\texttt{ct}' = \left(\left[\boldsymbol{c}_0^{(i)} + \left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_1^{(i)}), \overrightarrow{\texttt{evk}}_0^i \right\rangle\right]_q, \left[\left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_1^{(i)}), \overrightarrow{\texttt{evk}}_1^i \right\rangle\right]_q\right)$$

can be decrypted using the key $\boldsymbol{s}$. Indeed one can verify that:

$$\boldsymbol{c}_0' + \boldsymbol{c}_1' \cdot \boldsymbol{s} = \Delta[\boldsymbol{m}^{(i)}]_t + \boldsymbol{v}^{(i)} + \left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_1^{(i)}), \vec{\boldsymbol{e}}_i \right\rangle \bmod q$$

Therefore the key-switching procedure adds a noise $\boldsymbol{v}_{switch} = \left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_1^{(i)}), \vec{\boldsymbol{e}}_i \right\rangle$ of same size than the one of the relinearization procedure.

$$\|\boldsymbol{v}_{switch}\|_\infty \leq \frac{\delta_{\mathcal{R}} \ell_{\omega,q_i} \omega B_{err}}{2}$$

Remark 1.3.16. Making the evaluation key, or even the relinearization key, public implies to use a circular security assumption i.e. to assume that encrypting a secret key under its corresponding public key is secure.

A last thing to notice is that the action of $\mathcal{G}$ on a ciphertext applies to the encrypted message and the noise. Therefore it would be interesting to know by how much it increases the initial noise. The easiest way to get a general bound on this growth is to consider the infinite norm of the elements through the canonical embedding like the authors of [GHS12a] do. For $\boldsymbol{a} \in \mathcal{R}$ and ζ_m a primitive m-th root of unity it is defined as:

$$\|\boldsymbol{a}\|_\infty^{can} = \max_{\substack{1 \leq k < m \\ k \wedge m = 1}} \left(\boldsymbol{a}(\zeta_m^k)\right) = \|\sigma(\boldsymbol{a})\|_\infty \tag{1.3.12}$$

where σ denotes the canonical embedding map. Indeed since elements of $\mathcal{G}$ permute the roots of $\boldsymbol{\Phi}_m$, the canonical norm is invariant under the action of $\mathcal{G}$. If we consider the invertible Vandermonde matrix $\texttt{CRT}_m \in M_{n,n}(\mathbb{C})$ given by the $n = \varphi(m)$ primitive m-th roots of unity $(\zeta_{m,1}, \ldots, \zeta_{m,n})$:

$$\texttt{CRT}_m = \begin{pmatrix} 1 & \zeta_{m,1} & \cdots & \zeta_{m,1}^{n-1} \\ 1 & \zeta_{m,2} & \cdots & \zeta_{m,2}^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \zeta_{m,n} & \cdots & \zeta_{m,n}^{n-1} \end{pmatrix} \tag{1.3.13}$$

we can write for any $\boldsymbol{a} \in \mathcal{K}_m$:

$$\begin{aligned} \|\boldsymbol{a}\|_\infty &= \left\|\mathtt{CRT}_m^{-1}\sigma(\boldsymbol{a})\right\|_\infty \\ &\leq \left\|\mathtt{CRT}_m^{-1}\right\|_\infty \|\sigma(\boldsymbol{a})\|_\infty \\ &\leq \left\|\mathtt{CRT}_m^{-1}\right\|_\infty \|\boldsymbol{a}\|_\infty^{can} \\ &\leq c_m \|\boldsymbol{a}\|_\infty^{can} \end{aligned} \tag{1.3.14}$$

where $c_m = \|\mathtt{CRT}_m^{-1}\|_\infty$, with $\|A\|_\infty = \max\limits_{1\leq i\leq n}\left\{\sum\limits_{j=1}^{n}|a_{i,j}|\right\}$ for a square matrix $A \in M_{n,n}(\mathbb{C})$. The interested reader can find details about the constant c_m in [DPSZ12], we just would like to mention that it can grow super-polynomially with m and that when m is a power-of-two it is equal to 1. Thus going back to key-switching, we can write:

$$\left\|\boldsymbol{v}^{(i)}\right\|_\infty \leq c_m \|\boldsymbol{v}^{(i)}\|_\infty^{can} = c_m \|\boldsymbol{v}\|_\infty^{can}$$

Thus if the noise $\boldsymbol{v}$ of a ciphertext $\mathtt{ct}$ is such that:

$$c_m \|\boldsymbol{v}\|_\infty^{can} < B_{dec} - \delta_{\mathcal{R}} \ell_{\omega,q_i} \omega B_{err}/2 \tag{1.3.15}$$

then the ciphertext $\mathtt{ct}^{(i)} = (\boldsymbol{c}_0^{(i)}, \boldsymbol{c}_1^{(i)})$ will decrypt correctly after the key-switching procedure.

Chapter 2

Full RNS scaled-invariant schemes

Contents

This work focuses on practical improvement of scaled-invariant homomorphic schemes. The improvements will be highlighted on the FV scheme although other schemes like Yet Another Somewhat Homomorphic Encryption' (YASHE') [BLLN13], despite the fact that its security has been called into question recently [ABD16], could also benefit from this work. As mentioned in section 1.3.1, choosing the ciphertexts modulus q as a product of small moduli fitting with practical hardware requirements (machine word, etc), allows to use RNS representation for the coefficients and thus avoid the need of multi-precision arithmetic in almost the whole scheme. However, since RNS is hardly compatible with several operations: non-exact division, rounding and decomposition in basis ω occurring in decryption, multiplication and relinearization, it is required to switch to a positional system at some point for performing these operations.

In this work we show how to efficiently avoid any switch between RNS and a positional system for performing these operations. Therefore we avoid the costs associated to the conversions (cf. Figure 3) together with costly multi-precision arithmetic. We present our full RNS variant of FV and analyze the new bounds on noise growth, finally a software implementation highlights the practical benefits of our variant. It is important to note that this work is related to the arithmetic at the coefficient level, thus the security features of the original scheme are not modified. For the same reasons, we will not complicate futilely our exposition with polynomial arithmetic in the general case and will only consider the particular case of power-of-two cyclotomics $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ with NTT using the negative wrapped convolution technique (cf. section 1.3.2).

Preliminaries and notations. The ciphertexts modulus q is chosen as a product of k prime moduli $q_1 \cdots q_k$. Ciphertexts will be managed as polynomials (of degree 1) in $\mathcal{R}[Y]$. For $\mathtt{ct} \in \mathcal{R}[Y]$, we note $\|\mathtt{ct}\|_\infty = \max_i \|\mathtt{ct}[i]\|_\infty$, $\mathtt{ct}[i]$ being the coefficient of degree i in Y. The multiplicative law of $\mathcal{R}[Y]$ is denoted by $\star$. In the case of power-of-two cyclotomics the expansion factor $\delta_\mathcal{R} = \sup\{\|\boldsymbol{f} \cdot \boldsymbol{g}\|_\infty / \|\boldsymbol{f}\|_\infty \cdot \|\boldsymbol{g}\|_\infty : (\boldsymbol{f}, \boldsymbol{g}) \in (\mathcal{R} \setminus \{\mathbf{0}\})^2\}$ is equal to n. For our subsequent discussions on decryption and homomorphic multiplication, we denote the "Division and Rounding" in $\mathcal{R}[Y]$, depending on parameters t and q, by:

$$\mathtt{DR}_{t,q} : \mathtt{ct} = \sum_{j=0}^{\deg(\mathtt{ct})} \mathtt{ct}[j] Y^j \in \mathcal{R}[Y] \mapsto \sum_{j=0}^{\deg(\mathtt{ct})} \left\lfloor \frac{t}{q} \mathtt{ct}[j] \right\rceil Y^j \in \mathcal{R}[Y]. \tag{2.0.1}$$

The letter ν will denote the size of a machine word. Therefore, from now on, any modulus b (should it belong to basis q or any other RNS basis) is assumed to verify $b < 2^\nu$. In RNS, an "inner modular multiplication" (`IMM`) in a small ring like $\mathbb{Z}/b\mathbb{Z}$ is a core operation. If `EM` stands for an elementary multiplication of two words, in practice an `IMM` is costlier than an `EM`. But it can be well controlled. For instance, the moduli provided in `NFLlib` library [AMBG+16] (cf. Sect. 2.4) enable a modular reduction which reduces to one `EM` followed by a multiplication modulo 2^ν. Furthermore, the cost of an inner reduction can be limited by using lazy reduction, e.g. during RNS base conversions used throughout this paper. For the sake of

simplicity, NTT and invNTT will denote the Number Theoretic Transform and its inverse used with the negative wrapped-convolution technique in a ring $\mathcal{R}_b$ for a modulus b.

2.1 Towards a full RNS decryption

This section deals with the creation of a variant of the original decryption function $\texttt{Dec}_{\texttt{FV}}$, which will only involve RNS representation. The definition of $\texttt{Dec}_{\texttt{FV}}$ is recalled hereafter.

$$\texttt{Dec}_{\texttt{FV}}(\texttt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q[Y]) = \left[\texttt{DR}_{t,q}\left([\texttt{ct}(\boldsymbol{s})]_q\right)\right]_t = \left[\left\lfloor \frac{t}{q}[\boldsymbol{c}_0 + \boldsymbol{c}_1\boldsymbol{s}]_q \right\rfloor\right]_t$$

We recall that the idea is to compute $[\boldsymbol{c}_0+\boldsymbol{c}_1\boldsymbol{s}]_q = [\Delta[\boldsymbol{m}]_t+\boldsymbol{v}]_q$ to reveal the noise $\boldsymbol{v}$. If this noise is small enough, and given that $[\boldsymbol{m}]_t$ has been scaled by $\Delta = \lfloor q/2t \rfloor$, the function $\texttt{DR}_{t,q}$ allows to cancel the noise while scaling down $\Delta[\boldsymbol{m}]_t$ to recover $[\boldsymbol{m}]_t$. Concretely, decryption is correct as long as $\|\boldsymbol{v}\|_\infty < B_{dec} = (\Delta - |q|_t)/2$, i.e. the size of the noise should not go over this bound after homomorphic operations.

The division-and-rounding operation makes $\texttt{Dec}_{\texttt{FV}}$ hardly compatible with RNS at a first sight. Because RNS is of non positional nature, only exact integer division can be naturally performed (by multiplying by a modular inverse). But it is not the case here. And the rounding operation involves comparisons which require to switch from RNS to another positional system anyway, should it be a classical binary system or a mixed-radix one [ST67]. To get an efficient RNS variant of $\texttt{Dec}_{\texttt{FV}}$, we use an idea of [BEMP15]. To this end, we quickly recall relevant RNS tools.

2.1.1 Fast RNS base conversion

At some point, the decryption requires, among others, a polynomial to be converted from $\mathcal{R}_q$ to $\mathcal{R}_t$. To achieve such kind of operations as efficiently as possible, we suggest to use a "fast base conversion". In order to convert residues of $\boldsymbol{a} \in \mathcal{R}_q$ from base q to a base $\mathcal{B}$ (e.g. $\{t\}$) coprime to q, we compute:

$$\texttt{FastBconv}(\boldsymbol{a}, q, \mathcal{B}) = \left(\sum_{i=1}^{k} \left|\boldsymbol{a}\frac{q_i}{q}\right|_{q_i} \times \frac{q}{q_i} \bmod b\right)_{b\in\mathcal{B}}. \tag{2.1.1}$$

This conversion is relatively fast. This is because the sum should ideally be reduced modulo q in order to provide the exact value $\boldsymbol{a}$; instead, (2.1.1) provides $\boldsymbol{a} + \boldsymbol{\alpha}(\boldsymbol{a}_q)q$ for a $\boldsymbol{\alpha}(\boldsymbol{a}_q) \in \mathcal{R}$ with coefficients in $[0, k) \cap \mathbb{Z}$ (cf. section 1.3.1). Computing the coefficients of $\boldsymbol{\alpha}(\boldsymbol{a}_q)$ would require extra operations in RNS. So this step is by-passed, at the cost of an approximate result.

2.1.2 Approximate RNS rounding

The above mentioned fast conversion allows us to efficiently compute an approximation of $\lfloor \frac{t}{q}[\boldsymbol{c}_0 + \boldsymbol{c}_1\boldsymbol{s}]_q \rceil$ modulo t. The next step consists in correcting this approximation.

First, we remark that $|\mathtt{ct}(\boldsymbol{s})|_q$ can be used instead of $[\mathtt{ct}(\boldsymbol{s})]_q$. Indeed, the difference between these two polynomials is a multiple of q. So, the division-and-rounding turns it into a polynomial multiple of t, which is canceled by the last reduction modulo t. Second, a rounding would involve, at some point, a comparison. This is hardly compatible with RNS, so it is avoided. Therefore, we propose to simplify the computation, albeit at the price of possible errors, by replacing rounding by flooring. To this end, we use the following formula:

$$\left\lfloor \frac{t}{q} |\mathtt{ct}(\boldsymbol{s})|_q \right\rfloor = \frac{t|\mathtt{ct}(\boldsymbol{s})|_q - |t \cdot \mathtt{ct}(\boldsymbol{s})|_q}{q}.$$

The division is now exact, so it can be done in RNS. Since this computation has to be done modulo t, the term $t|\mathtt{ct}(\boldsymbol{s})|_q$ cancels. Furthermore, the term $(|t \cdot \mathtt{ct}(\boldsymbol{s})|_q \bmod t)$ can be obtained through a fast conversion. Lemma 2.1.1 sums up the strategy by replacing $|\mathtt{ct}(\boldsymbol{s})|_q$ by $\gamma|\mathtt{ct}(\boldsymbol{s})|_q$, where γ is an integer which will help in correcting the approximation error.

Lemma 2.1.1. Let $\mathtt{ct}$ be such that $[\mathtt{ct}(\boldsymbol{s})]_q = \Delta[\boldsymbol{m}]_t + \boldsymbol{v} + q\boldsymbol{r}$, and let $\boldsymbol{v}_c := t\boldsymbol{v} - [\boldsymbol{m}]_t|q|_t$. Let γ be an integer coprime to q. Then, for $b \in \{t, \gamma\}$, the following equalities hold modulo b:

$$\begin{aligned} \mathtt{FastBconv}(|\gamma t \cdot \mathtt{ct}(\boldsymbol{s})|_q, q, \{t, \gamma\}) \times |-q^{-1}|_b &= \left\lfloor \gamma \frac{t}{q} [\mathtt{ct}(\boldsymbol{s})]_q \right\rfloor - \boldsymbol{e} \\ &= \gamma\left([\boldsymbol{m}]_t + t\boldsymbol{r}\right) + \left\lfloor \gamma \frac{\boldsymbol{v}_c}{q} \right\rfloor - \boldsymbol{e} \end{aligned} \tag{2.1.2}$$

where each integer coefficient of the error polynomial $\boldsymbol{e} \in \mathcal{R}$ lies in $[0, k]$.

Proof. According to (2.1.1), $\mathtt{FastBconv}(|t\gamma \cdot \mathtt{ct}(\boldsymbol{s})|_q, q, \{t, \gamma\})$ provides $|t\gamma \cdot \mathtt{ct}(\boldsymbol{s})|_q + q\boldsymbol{a}$, where each coefficient $\boldsymbol{a}_i$ is an integer lying in $[0, k) \cap \mathbb{Z}$. Let b be t or γ. Then,

$$\begin{aligned} \mathtt{FastBconv}(|t\gamma \cdot \mathtt{ct}(\boldsymbol{s})|_q, q, \{t, \gamma\}) \times |-q^{-1}|_b &= (|t\gamma \cdot \mathtt{ct}(\boldsymbol{s})|_q + q\boldsymbol{a}) \times |-q^{-1}|_b \bmod b \\ &= \frac{t\gamma[\mathtt{ct}(\boldsymbol{s})]_q - |t\gamma \cdot \mathtt{ct}(\boldsymbol{s})|_q - q\boldsymbol{a}}{q} \bmod b \\ &= \left\lfloor \frac{t\gamma[\mathtt{ct}(\boldsymbol{s})]_q}{q} \right\rfloor - \boldsymbol{a} \bmod b \\ &= \left\lfloor \frac{t\gamma[\mathtt{ct}(\boldsymbol{s})]_q}{q} \right\rfloor - \boldsymbol{e} \bmod b \end{aligned}$$

where $\boldsymbol{e} = \boldsymbol{a} + \boldsymbol{u}$, with $u_i \in \{0, 1\}$, i.e. $\boldsymbol{e}_i \in [0, k] \cap \mathbb{Z}$. To conclude the proof, it suffices to use the equality $\Delta t = q - |q|_t$. That way, one can write:

$$t[\mathtt{ct}(\boldsymbol{s})]_q = q([\boldsymbol{m}]_t + t\boldsymbol{r}) + \boldsymbol{v}_c$$

and the second equality (2.1.2) follows. □

The error $\boldsymbol{e}$, due to the fast conversion and the replacement of rounding by flooring, is the same error for residues modulo t and γ. The residues modulo γ will enable a fast correction

of it and of the term $\lfloor \gamma \frac{v_c}{q} \rceil$ at a same time. Also, note that r vanishes since it is multiplied by both t and γ.

2.1.3 Correcting the approximate RNS rounding

The next step is to show how γ can be used to correct the term $(\lfloor \gamma \frac{v_c}{q} \rceil - e)$ of (2.1.2). In practice it can be done efficiently when the polynomial v_c is such that $\|v_c\|_\infty \leqslant q(\frac{1}{2} - \varepsilon)$, for some real number $\varepsilon \in (0, 1/2]$.

Lemma 2.1.2. Let $\|v_c\|_\infty \leqslant q(\frac{1}{2}-\varepsilon)$, $e \in \mathcal{R}$ with coefficients in $[0, k]$, and γ an integer. Then,

$$\gamma\varepsilon \geqslant k \Rightarrow \left[\left\lfloor \gamma \frac{v_c}{q} \right\rceil - e \right]_\gamma = \left\lfloor \gamma \frac{v_c}{q} \right\rceil - e. \tag{2.1.3}$$

Proof. By hypothesis, the coefficients $\left\lfloor \gamma \frac{(v_c)_i}{q} \right\rceil - e_i$ belong to $\left[-\gamma(\frac{1}{2} - \varepsilon) - k, \gamma(\frac{1}{2} - \varepsilon) \right] \cap \mathbb{Z}$ for any $i \in [0, n) \cap \mathbb{Z}$. To have $[\lfloor \gamma \frac{v_c}{q} \rceil - e]_\gamma = \lfloor \gamma \frac{v_c}{q} \rceil - e$, we need to ensure:

$$-\left\lfloor \frac{\gamma}{2} \right\rfloor - \frac{1}{2} \leqslant \gamma \frac{(v_c)_i}{q} - e_i < \left\lfloor \frac{\gamma - 1}{2} \right\rfloor + \frac{1}{2}$$

A sufficient condition for this is given by:

$$\begin{cases} \gamma\left(\frac{1}{2} - \varepsilon\right) & < & \left\lfloor \frac{\gamma-1}{2} \right\rfloor + \frac{1}{2} \\ -\left\lfloor \frac{\gamma}{2} \right\rfloor - \frac{1}{2} & \leqslant & -\gamma\left(\frac{1}{2} - \varepsilon\right) - k \end{cases} \Leftrightarrow \begin{cases} (\gamma \text{ odd}) & \begin{cases} \gamma\varepsilon & > & 0 \\ \gamma\varepsilon & \geqslant & k \end{cases} \\ (\gamma \text{ even}) & \begin{cases} \gamma\varepsilon & > & \frac{1}{2} \\ \gamma\varepsilon & \geqslant & k - \frac{1}{2} \end{cases} \end{cases}$$

□

Lemma 2.1.2 enables an efficient and correct RNS rounding as long as γ, which has to be greater than:

$$\gamma \geq k\varepsilon^{-1} = k\left(\frac{1}{2} - \frac{\|v_c\|_\infty}{q}\right)^{-1}$$

has the size of a modulus i.e. such that $\gamma < 2^\nu$. Concretely, one computes (2.1.2) and uses the centered remainder modulo γ to obtain $\gamma\,([m]_t + tr)$ modulo t, which reduces to $\gamma[m]_t \bmod t$. In the end, it remains to multiply by $|\gamma^{-1}|_t$ to recover $[m]_t$.

2.1.4 A full RNS variant of FV decryption

The new variant of the decryption is detailed in Algorithm 14. The main modification for the proposed RNS decryption is due to lemma 2.1.2. As stated by theorem 2.1.3, for a given γ, the correctness of rounding requires a new bound on the noise to make the "γ-correction" technique successful.

Theorem 2.1.3. Let $\mathtt{ct}(s) = \Delta[m]_t + v \pmod q$. Let γ be a positive integer coprime to t and q such that $\gamma > 2k/\left(1 - \frac{t|q|_t}{q}\right)$. For Algorithm 14 returning $[m]_t$, it suffices that v satisfies

the following bound:

$$\|\boldsymbol{v}\|_\infty \leqslant \frac{q}{t}\left(\frac{1}{2} - \frac{k}{\gamma}\right) - \frac{|q|_t}{2}. \tag{2.1.4}$$

Proof. According to lemma 2.1.2, the γ-correction technique works as long as:

$$\gamma\left(\frac{1}{2} - \frac{\|\boldsymbol{v_c}\|_\infty}{q}\right) \geqslant k \Leftrightarrow \|\boldsymbol{v_c}\|_\infty \leqslant q\left(\frac{1}{2} - \frac{k}{\gamma}\right)$$

Moreover since we have:

$$\|\boldsymbol{v_c}\|_\infty = \|t\boldsymbol{v} - |q|_t[\boldsymbol{m}]_t\|_\infty \leqslant t\|\boldsymbol{v}\|_\infty + |q|_t\frac{t}{2}$$

The bound (2.1.4) follows, and the lower bound on γ just ensures it is positive. □

There is a trade-off between the size of γ and the bound in (2.1.4). From a computational point of view γ should ideally be as small as possible ($\gamma \simeq 2k$) but it would result in a quite small bound for the noise. However by choosing $\gamma \simeq 2^{p+1}k$ for $p < \nu - 1 - \lceil \log_2(k) \rceil$ (i.e. $\gamma < 2^\nu$ is a standard modulus), the bound $(\Delta(1-2^{-p}) - |q|_t)/2$ for a correct decryption should be close to the original bound $(\Delta - |q|_t)/2$ for practical values of ν. In section 2.4.1 we will provide a concrete estimation of γ which shows that γ can be chosen very close to $2k$ in practice, and thus fitting in a basic machine word by far.

Algorithm 14 RNS Decryption of FV

Require: a ciphertext $\mathtt{ct}$, encrypting a message $\boldsymbol{m} \in \mathcal{R}_t$ and whose noise $\boldsymbol{v}$ satisfies (2.1.4), and the secret key $\boldsymbol{s}$ both in base q, an integer γ satisfying hypothesis of theorem 2.1.3
Ensure: $[\boldsymbol{m}]_t$

```
function DEC_RNS(ct, s, γ))
    for b ∈ {t, γ} do
        s^(b) ← FastBconv(|γt · ct(s)|_q, q, {b}) × | − q^{-1}|_b mod b
    s̃^(γ) ← [s^(γ)]_γ
    m^(t) ← [(s^(t) − s̃^(γ)) × |γ^{-1}|_t]_t
    return m^(t)
```

2.1.5 Staying in RNS is asymptotically better

In the decryption technique, $(\mathtt{ct}(\boldsymbol{s}) \bmod q)$ has to be computed first. To optimize this polynomial product, one basically performs $k\mathtt{NTT} \to kn\mathtt{IMM} \to k\mathtt{invNTT}$. For the following steps, a simple strategy is to compute $(\left\lfloor \frac{t}{q}[\mathtt{ct}(\boldsymbol{s})]_q \right\rceil \bmod t)$ by doing an RNS-to-binary conversion in order to perform the division and rounding. By denoting $\boldsymbol{x}_i = \left|\mathtt{ct}(\boldsymbol{s})\frac{q_i}{q}\right|_{q_i}$, one computes $\sum_{i=1}^{k} \boldsymbol{x}_i \frac{q}{q_i} \bmod q$, compares it to $q/2$ so as to center the result, and performs division and rounding. Since the $\boldsymbol{x}_i$s fit in one word and the q/q_is fit in $k-1$ words, each product $\boldsymbol{x}_i q/q_i$ requires $n(k-1)\mathtt{EM}$. The scaling and rounding only requires $O(n)$ multiplications, hence the whole procedure requires $O(k^2 n)\mathtt{EM}$. As shown in equation (1.1.3), in practice security analysis (cf. e.g. [FV12], [BLLN13], [LN14]) requires that $k\nu = \lceil \log_2(q) \rceil \in O(n)$ which leads to

an asymptotic complexity of $O(n^3)\texttt{EM}$ for leaving RNS representation. On the other hand, the cost of evaluating the ciphertext on the secret key is asymptotically $O(kn\log_2(n))\texttt{EM}$, i.e. $O(n^2\log_2(n))\texttt{EM}$, since $k \in O(n)$. So, the cost of leaving RNS to access a positional system is dominant in the asymptotic computational complexity.

Staying in RNS enables to get a better asymptotic complexity. Indeed, since the operations in Algorithm 14 are always made modulo $b \in \{t, \gamma\}$, all the elements fit in one word, therefore it requires $O(kn)\texttt{EM}$ operations (excluding the polynomial product). Thus, the cost of $\texttt{NTT}$ is dominant in this case. By considering $k \in O(n)$, we deduce $C(\texttt{Dec}_{\texttt{FV}}) \in O(n^3)$, while $C(\texttt{Dec}_{\texttt{RNS}}) \in O(n^2\log_2(n))$. However, the hidden constant in "$k \in O(n)$" is small, and the $\texttt{NTT}$, common to both variants, should avoid any noticeable divergence (cf. 2.4.4) for practical ranges for parameters.

In order to provide optimized RNS variants of decryption, we make two remarks.

- First, the reduction modulo q is unnecessary. Indeed, any extra multiple of q in the sum $\sum_{i=1}^{k} \boldsymbol{x}_i \frac{q}{q_i}$ is multiplied by $\frac{t}{q}$, making the resulting term a multiple of t, which is not affected by the rounding and is finally canceled modulo t.

- Second, it is possible to precompute $\frac{t}{q}$ as a multi-precision floating point number in order to avoid a costly integer division. But given the first remark, it suffices to precompute the floating point numbers $\boldsymbol{Q}_i \sim \frac{t}{q_i}$ with $2\nu+\log_2(k)-\log_2(t)$ bits (~ 2 words) of precision. In this case, using standard double or quadruple (depending on ν) precision is sufficient. Finally, it is sufficient to compute $\lfloor \sum_{i=1}^{k} \boldsymbol{x}_i\boldsymbol{Q}_i \rceil \bmod t$. This represents about $2kn\texttt{EM}$, in particular since reducing modulo t is nearly free of cost when t is a power of 2.

Actually, we can perform the computations even more efficiently. In Algorithm 14, γ is assumed to be coprime to t. It is possible to be slightly more efficient by noticing that the coprimality assumption can be avoided since the division by γ is exact. To do it, the $\texttt{for}$ loop can be done modulo $\gamma \times t$. For instance, even if t a power of 2, one can choose γ as being a power of 2, and thanks to the following lemma, finish the decryption very efficiently.

Lemma 2.1.4. Let γ be a power of 2. Let $\boldsymbol{z} := |\gamma[\boldsymbol{m}]_t + \lfloor \gamma\frac{\boldsymbol{v}_c}{q} \rceil - \boldsymbol{e}|_{\gamma t}$ coming from (2.1.2) when computed modulo γt. If γ satisfies (2.1.3), then ($\gg$ denotes the right bit-shifting, and & the bit-wise $\texttt{and}$)

$$[(\boldsymbol{z} + (\boldsymbol{z}\&(\gamma-1))) \gg \log_2(\gamma)]_t = [\boldsymbol{m}]_t. \tag{2.1.5}$$

Proof. Let's denote $\tilde{\boldsymbol{v}}_c := \lfloor \gamma\frac{\boldsymbol{v}_c}{q} \rceil - \boldsymbol{e}$. By computing (2.1.2) modulo γt, we obtain:

$$\boldsymbol{z} = |\gamma[\boldsymbol{m}]_t + \tilde{\boldsymbol{v}}_c|_{\gamma t}$$

First, we notice that we can also write $\boldsymbol{z} = |\gamma|\boldsymbol{m}|_t + \tilde{\boldsymbol{v}}_c|_{\gamma t}$. Indeed, $\gamma|\boldsymbol{m}|_t = \gamma([\boldsymbol{m}]_t + t\boldsymbol{a})$, where $\boldsymbol{a}_i \in \{0, 1\}$. Thus, $\gamma t\boldsymbol{a}$ vanishes modulo γt. Next, for any t, and because γ is a power of 2, we

have $\boldsymbol{z}\&(\gamma-1) = |\boldsymbol{z}|_\gamma = |\tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_\gamma$. Consequently,

$$\boldsymbol{z} + \boldsymbol{z}\&(\gamma-1) = |\gamma|\boldsymbol{m}|_t + \tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_{\gamma t} + |\tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_\gamma$$

Now, (2.1.3) means that γ is chosen such that the coefficients $(\tilde{\boldsymbol{v}}_{\boldsymbol{c}})_i$, $i \in \{0,\dots,n-1\}$, lie in $[-\frac{\gamma}{2}, \frac{\gamma}{2})$. This, together with the fact that $(\gamma|\boldsymbol{m}|_t)_i \in [0, \gamma(t-1)]$, implies that we can write $|\gamma|\boldsymbol{m}|_t + \tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_{\gamma t} = \gamma|\boldsymbol{m}|_t + \tilde{\boldsymbol{v}}_{\boldsymbol{c}} + \gamma t\boldsymbol{b}$ with $\boldsymbol{b}_i \in \{0,1\}$ ($\boldsymbol{b}_i = 1 \Leftrightarrow ((\gamma|\boldsymbol{m}|_t)_i = 0$ and $(\tilde{\boldsymbol{v}}_{\boldsymbol{c}})_i < 0)$). To sum up, we have established so far that:

$$\boldsymbol{z} + \boldsymbol{z}\&(\gamma-1) = \gamma|\boldsymbol{m}|_t + \tilde{\boldsymbol{v}}_{\boldsymbol{c}} + |\tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_\gamma + \gamma t\boldsymbol{b}$$

The next step is to show that any coefficient of $\tilde{\boldsymbol{v}}_{\boldsymbol{c}} + |\tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_\gamma$ lies in $[0,\gamma)$. This is a direct consequence of the fact that $(\tilde{\boldsymbol{v}}_{\boldsymbol{c}})_i \in [-\frac{\gamma}{2}, \frac{\gamma}{2})$. Indeed, we have:

$$\forall i \in [0,n-1] \quad , \begin{cases} (\tilde{\boldsymbol{v}}_{\boldsymbol{c}})_i \in [-\gamma/2, 0) \\ (\tilde{\boldsymbol{v}}_{\boldsymbol{c}})_i \in [0, \gamma/2) \end{cases} \Rightarrow \begin{cases} \left(|\tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_\gamma\right)_i = (\tilde{\boldsymbol{v}}_{\boldsymbol{c}})_i + \gamma \\ \left(|\tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_\gamma\right)_i = (\tilde{\boldsymbol{v}}_{\boldsymbol{c}})_i \end{cases}$$

$$\Rightarrow \begin{cases} \left(\tilde{\boldsymbol{v}}_{\boldsymbol{c}} + |\tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_\gamma\right)_i \in [0, \gamma-2] \\ \left(\tilde{\boldsymbol{v}}_{\boldsymbol{c}} + |\tilde{\boldsymbol{v}}_{\boldsymbol{c}}|_\gamma\right)_i \in [0, \gamma-2] \end{cases}$$

Consequently, $(\boldsymbol{z} + \boldsymbol{z}\&(\gamma-1)) \gg \log_2(\gamma) = |\boldsymbol{m}|_t + t\boldsymbol{b}$, and (2.1.5) follows. □

Lemma 2.1.4 can be adapted to other values for γ, but choosing it as a power of 2 makes the computation very easy because it is composed of simple operations on bits. Finally, as soon as γt fits in 1 word, the cost of such variant (besides the polynomial product) reduces to knIMM, or simply to knEM modulo $2^{\log_2(\gamma t)}$ whenever t is a power of 2.

Remark 2.1.5. In the previous discussion, the product γt was assumed to fit in one machine word to simplify complexity analysis. However, for some applications, the plaintext modulus t can be bigger than a machine word (e.g. homomorphic neural networks [GDL+16], where $t > 2^{80}$). In such cases, either the plaintexts directly lie in $\mathcal{R}_t$, or t can be decomposed in a product of smaller moduli $t_1,\dots,t_\ell$, enabling the use of RNS for encoding plaintexts (and then allowing better homomorphic multiplicative depth for a given dimension n). In the first case, the optimized RNS decryption (given by lemma 2.1.4) remains available, but the residues modulo t should be handled with several words. In the second case, a plaintext is recovered by decrypting its residues modulo each of the t_i. These ℓ decryptions can be done as in lemma 2.1.4, by using γ as a power of 2 (whatever the t_i's are). Finally, the plaintext is reconstructed from residues modulo the t_i's by using a classical RNS to binary conversion. However, this conversion is only related to the way the plaintexts are encoded, hence it does not affect the RNS description described in this section which essentially deals with arithmetic of the ciphertexts, i.e. modulo q.

2.2 Towards a full RNS homomorphic multiplication

Assume that we want to multiply homomorphically two degree 1 ciphertexts $\mathtt{ct}_1$ and $\mathtt{ct}_2$. The main obstacle to a full RNS variant of the multiplication procedure in the original FV is the call to $\mathtt{DR}_{t,q}$ for the scaling of the degree 2 ciphertext $\mathtt{ct}_\star = \mathtt{ct}_1 \star \mathtt{ct}_2 \in \mathcal{R}^3$. Indeed, the context here is different than for the decryption. While in the decryption we are working with a noise whose size can be controlled, and while we are reducing a value from q to $\{t\}$, here the polynomial coefficients of the product $\mathtt{ct}_1 \star \mathtt{ct}_2$ have kind of random size modulo q (for each integer coefficient) and have to be reduced towards q. Roughly the idea is to use a second RNS basis $\mathcal{B}$ in which we can compute the scaling by t/q and perform the rounding. We add two moduli to $\mathcal{B}$: one modulus $\tilde{b}$ to reduce the q-overflow due to the first fast conversion from q to $\mathcal{B}$ and one modulus b_{sk} which is used to perform an exact conversion from bases $\mathcal{B}$ to q. We summarize our strategy in Figure 4.

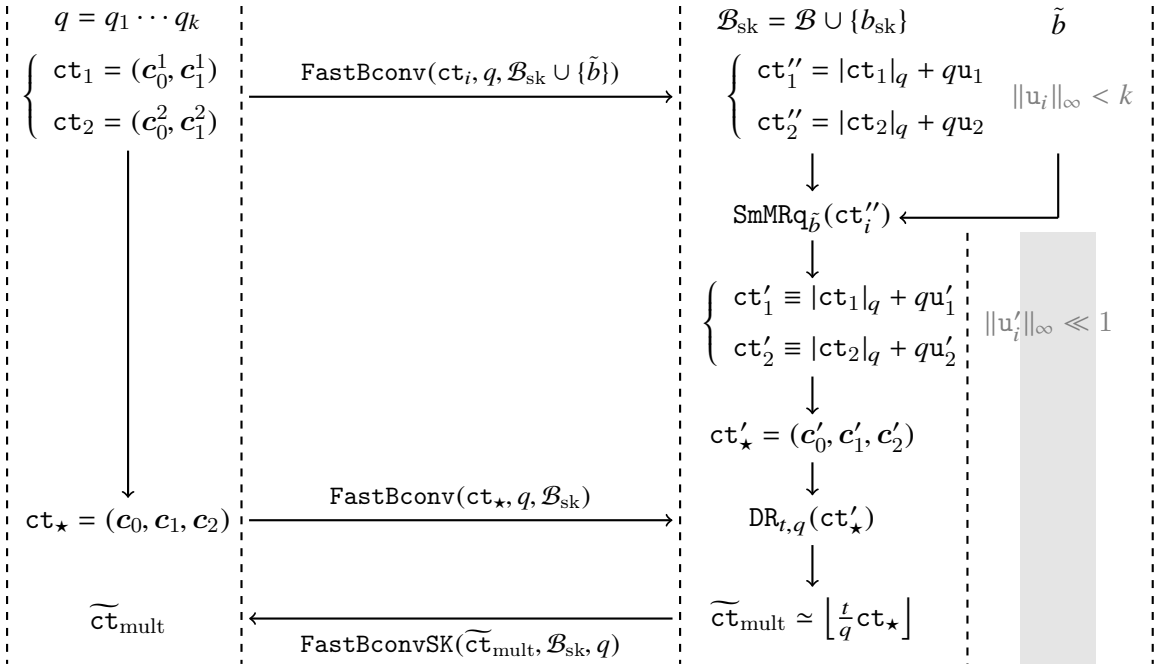


Figure 4: Overview of the strategy for the homomorphic multiplication of FV in RNS

2.2.1 Auxiliary RNS bases

The computation of $\mathtt{ct}_\star = \mathtt{ct}_1 \star \mathtt{ct}_2 \in \mathcal{R}^3$, which is the first step of multiplication, requires to use enough moduli to contain any product, in $\mathcal{R}[Y]$ (coefficients in $\mathbb{Z}$), of degree-1 elements from $\mathcal{R}_q[Y]$ (coefficients modulo q). So, we need an auxiliary basis $\mathcal{B}$, additionally to the basis q. We assume that $\mathcal{B}$ contains ℓ moduli (while q has k elements). A sufficient size for ℓ will be given later. An extra modulus $b_{\mathtt{sk}}$ is added to $\mathcal{B}$ to create an extended base $\mathcal{B}_{\mathtt{sk}}$. It will be used for a transition between the new steps 1 and 2. Computing the residues of ciphertexts in $\mathcal{B}_{\mathtt{sk}}$ is done through a fast conversion from q (cf. Equation ((2.1.1))). In order to reduce the extra multiples of q (called "q-overflows" in further discussions) this conversion can produce,

a single-modulus base $\tilde{b}$ is introduced. All these bases are assumed to be pairwise coprime which means that all the moduli are pairwise coprime.

Reducing (mod q) a ciphertext in $\mathcal{B}_{\mathtt{sk}}$ Prior to each multiplication we need the residues of the degree 1 ciphertexts $\mathtt{ct}_1$ and $\mathtt{ct}_2$ in $q \cup \mathcal{B}_{\mathtt{sk}}$, therefore the residues in basis q must be extended to $\mathcal{B}_{\mathtt{sk}}$. However, fast conversions (cf. Section 2.1.1) from q can create q-overflows (i.e. unnecessary multiples of q) in the output which would be added to the noise. Therefore to limit the impact on noise growth, we give an efficient way to reduce a polynomial $\boldsymbol{c}'' = \boldsymbol{c}+q\boldsymbol{u}$ in $\mathcal{B}_{\mathtt{sk}}$. For that purpose, we need to add an extra modulus $\tilde{b}$ to $\mathcal{B}_{\mathtt{sk}}$. The strategy is based on a Montgomery reduction and is described in Algorithm 15.

Algorithm 15 Small Montgomery Reduction modulo q

Require: $\boldsymbol{c}''$ in $\mathcal{B}_{\mathtt{sk}} \cup \{\tilde{b}\}$
Ensure: $\boldsymbol{c}'$ in $\mathcal{B}_{\mathtt{sk}}$, with $\boldsymbol{c}' \equiv \boldsymbol{c}''\tilde{b}^{-1} \bmod q$, $\|\boldsymbol{c}'\|_\infty \leqslant \|\boldsymbol{c}''\|_\infty/\tilde{b} + q/2$
1 **function** $\mathtt{SmMRq}_{\tilde{b}}((\boldsymbol{c}''_m)_{b\in\mathcal{B}_{\mathtt{sk}}\cup\{\tilde{b}\}})$
2 $\quad \boldsymbol{r}_{\tilde{b}} \leftarrow [-\boldsymbol{c}''_{\tilde{b}}/q]_{\tilde{b}}$
3 $\quad$ **for** $b \in \mathcal{B}_{\mathtt{sk}}$ **do**
4 $\quad\quad \boldsymbol{c}'_b \leftarrow |(\boldsymbol{c}''_b + q\boldsymbol{r}_{\tilde{b}})\tilde{b}^{-1}|_b$
5 $\quad$ **return** $\boldsymbol{c}'$ in $\mathcal{B}_{\mathtt{sk}}$

Lemma 2.2.1. On input $\boldsymbol{c}''_b = |[\tilde{b}\boldsymbol{c}]_q + q\boldsymbol{u}|_b$ for all $b \in \mathcal{B}_{\mathtt{sk}} \cup \{\tilde{b}\}$, with $\|\boldsymbol{u}\|_\infty \leqslant \tau$, and given a parameter $\rho > 0$ such that:

$$\tilde{b}\rho \geqslant 2\tau + 1. \tag{2.2.1}$$

Algorithm 15 returns $\boldsymbol{c}'$ in $\mathcal{B}_{\mathtt{sk}}$ with $\boldsymbol{c}' \equiv \boldsymbol{c} \bmod q$ and $\|\boldsymbol{c}'\|_\infty \leqslant \frac{q}{2}(1+\rho)$.

Proof. Algorithm 15 performs in $\mathcal{B}_{\mathtt{sk}}$ the computation of:

$$\frac{[\boldsymbol{c}\tilde{b}]_q + q\boldsymbol{u} + q[-([\boldsymbol{c}\tilde{b}]_q + q\boldsymbol{u})/q]_{\tilde{b}}}{\tilde{b}}.$$

This quantity is clearly congruent to $\boldsymbol{c}$ modulo q. In accordance with hypothesis (2.2.1), its norm is bounded by:

$$\frac{q(1/2 + \tau + \tilde{b}/2)}{\tilde{b}} \leqslant \frac{q}{2}(1+\rho)$$

□

Remark 2.2.2. To use this fast reduction, the ciphertexts have to be handled in base q through the Montgomery [Mon85] representation with respect to $\tilde{b}$ (i.e. $|\tilde{b}\boldsymbol{c}|_q$ instead of $|\boldsymbol{c}|_q$). This can be done for free during the fast conversions, by multiplying the residues of $\boldsymbol{c}$ by precomputed $\left|\tilde{b}q_i/q\right|_{q_i}$ instead of $|q_i/q|_{q_i}$. Moreover, since $\{\tilde{b}\}$ is a single-modulus base, conversion of $\boldsymbol{r}_{\tilde{b}}$ from $\{\tilde{b}\}$ to $\mathcal{B}_{\mathtt{sk}}$ (line 4 of Algorithm 15) is a simple copy-paste when $\tilde{b} < b_i$. Finally, if Algorithm 15 is performed right after a fast extension from q (for converting $|\tilde{b}\boldsymbol{c}|_q$), τ is nothing but $k-1$ (cf. section 1.3.1).

Hence at this point we have the residues of both ciphertexts $\mathtt{ct}_1$ and $\mathtt{ct}_2$ in $q \cup \mathcal{B}_{\mathtt{sk}}$ (with the residues $\mathtt{ct}'_i$ ($\equiv \mathtt{ct}_i \bmod q$) in $\mathcal{B}_{\mathtt{sk}}$ reduced through Algorithm 15). Therefore we can compute the product $\mathtt{ct}_\star = \mathtt{ct}_1 \star \mathtt{ct}_2$ in q and $\mathtt{ct}'_\star = \mathtt{ct}'_1 \star \mathtt{ct}'_2$ in $\mathcal{B}_{\mathtt{sk}}$.

2.2.2 Adapting the scaling

We recall that originally this step is the computation of $\left\lceil \mathtt{DR}_{t,q}(\mathtt{ct}_\star) \right\rfloor_q$ (cf. Equation (2.0.1)). Unlike the decryption, a γ-correction technique does not guarantee an exact rounding. Indeed, for the decryption we wanted to get $\mathtt{DR}_{t,q}\left([\mathtt{ct}(\boldsymbol{s})]_q\right)$, and through $\boldsymbol{s}$ we had access to the noise of $\mathtt{ct}$, on which we had some control. In the present context, we cannot ensure a condition like $\|[t \cdot \mathtt{ct}_\star]_q\|_\infty \leqslant q(\frac{1}{2} - \varepsilon)$, for some $\varepsilon^{-1} \sim 2^\nu$, which would enable the use of an efficient γ-correction. Therefore, we suggest to perform a simple uncorrected RNS flooring. For that purpose, we define for $\boldsymbol{a} \in \mathcal{R}$ and $b \in \mathcal{B}_{\mathtt{sk}}$:

$$\mathtt{fastRNSFloor}_q(\boldsymbol{a}, b) := (\boldsymbol{a} - \mathtt{FastBconv}(|\boldsymbol{a}|_q, q, b)) \times |q^{-1}|_b \bmod b.$$

Since Algorithm 15 should be executed first, lemma 2.2.1 ensures that if $\tilde{b}$ satisfies the bound (2.2.1), for a given parameter $\rho > 0$, then the residues of $\mathtt{ct}'_i$ ($\equiv \mathtt{ct}_i \bmod q$) in $\mathcal{B}_{\mathtt{sk}}$ are such that:

$$\|\mathtt{ct}'_\star := \mathtt{ct}'_1 \star \mathtt{ct}'_2\|_\infty \leqslant \delta_{\mathcal{R}} \frac{q^2}{2} (1+\rho)^2 . \tag{2.2.2}$$

The parameter ρ is chosen as small as possible in practice to limit the noise growth but such that equation (2.2.1) holds. Notice that, in base q, $\mathtt{ct}'_i$ and $\mathtt{ct}_i$ are equals.

Lemma 2.2.3. Let the residues of $\mathtt{ct}'_i \equiv \mathtt{ct}_i \bmod q$ be given in base $q \cup \mathcal{B}_{\mathtt{sk}}$, with $\|\mathtt{ct}'_i\|_\infty \leqslant \frac{q}{2}(1+\rho)$ for $i \in \{1,2\}$. Let $\mathtt{ct}'_\star = \mathtt{ct}'_1 \star \mathtt{ct}'_2$. Then, for $j \in \{0,1,2\}$,

$$\mathtt{fastRNSFloor}_q(t \cdot \mathtt{ct}'_\star[j], \mathcal{B}_{\mathtt{sk}}) = \left\lfloor \frac{t}{q} \mathtt{ct}'_\star[j] \right\rceil + \boldsymbol{b}_j \text{ in } \mathcal{B}_{\mathtt{sk}}, \text{ with } \|\boldsymbol{b}_j\|_\infty \leqslant k. \tag{2.2.3}$$

Proof. We recall that $\mathtt{FastBconv}(|t \cdot \mathtt{ct}'_\star[j]|_q, q, \mathcal{B}_{\mathtt{sk}})$ outputs $|t \cdot \mathtt{ct}'_\star[j]|_q + q\boldsymbol{u}$ with $\|\boldsymbol{u}\|_\infty \leqslant k-1$. Then, the proof is complete by using the general equalities:

$$\frac{x - |x|_q}{q} = \left\lfloor \frac{x}{q} \right\rceil = \left\lfloor \frac{x}{q} \right\rfloor + \tau \quad \text{with } \tau \in \{-1, 0\}$$

□

Therefore we can use the residues in basis q to compute $\mathtt{DR}_{t,q}(\mathtt{ct}_\star)$, through $\mathtt{fastRNSFloor}$, in $\mathcal{B}_{\mathtt{sk}}$. Note that because of the division by q this cannot be done in basis q. Thus at this point we have the residues of $\widetilde{\mathtt{ct}}_{\mathtt{mult}} = \mathtt{DR}_{t,q}(\mathtt{ct}_\star)$ in $\mathcal{B}_{\mathtt{sk}}$ whose inherent noise is bounded by the following proposition.

Proposition 2.2.4. Let $\widetilde{\mathtt{ct}}_{\mathtt{mult}} = \mathtt{DR}_2(\mathtt{ct}'_\star)$ with $\mathtt{ct}'_\star$ satisfying (2.2.2), and

$$r_\infty := \frac{1+\rho}{2}(1+\delta_{\mathcal{R}} B_{key}) + 1$$

Let $\boldsymbol{v}_i$ be the inherent noise of $\mathtt{ct}'_i$. Then $\widetilde{\mathtt{ct}}_{\mathtt{mult}}(\boldsymbol{s}) = \Delta\,[\boldsymbol{m}_1\boldsymbol{m}_2]_t + \tilde{\boldsymbol{v}}_{\mathtt{mult}} \bmod q$ with:

$$\begin{aligned}\|\tilde{\boldsymbol{v}}_{\text{mult}}\|_\infty \leq & \delta_{\mathcal{R}} t \left(r_\infty + \frac{1}{2}\right)(\|\boldsymbol{v}_1\|_\infty + \|\boldsymbol{v}_2\|_\infty) + \frac{\delta_{\mathcal{R}}}{2}\min(\|\boldsymbol{v}_1\|_\infty, \|\boldsymbol{v}_2\|_\infty) \\ & + |q|_t \delta_{\mathcal{R}} t \left(r_\infty + \frac{3}{2}\right) + \frac{1+\delta_{\mathcal{R}} B_{key} + \delta_{\mathcal{R}}^2 B_{key}^2}{2}. \end{aligned} \tag{2.2.4}$$

Proof. This proof is similar to the one given in the first chapter. Some of the tools and bounds from there are re-used here. In the following, we write $\mathtt{ct}'_i = (\boldsymbol{c}'_{i,0}, \boldsymbol{c}'_{i,1})$. In particular, we have $\mathtt{ct}'_\star = (\boldsymbol{c}'_{1,0}\cdot\boldsymbol{c}'_{2,0}, \boldsymbol{c}'_{1,1}\cdot\boldsymbol{c}'_{2,0} + \boldsymbol{c}'_{1,0}\cdot\boldsymbol{c}'_{2,1}, \boldsymbol{c}'_{1,1}\cdot\boldsymbol{c}'_{2,1})$. By hypothesis, each $\boldsymbol{c}'_{i,j}$ satisfies $\|\boldsymbol{c}'_{i,j}\|_\infty \leqslant \frac{q}{2}(1+\rho)$. In particular, the bound in (2.2.2) comes from the fact that:

$$\|\boldsymbol{c}'_{1,1}\cdot\boldsymbol{c}'_{2,0} + \boldsymbol{c}'_{1,0}\cdot\boldsymbol{c}'_{2,1}\|_\infty \leqslant \delta_{\mathcal{R}}\frac{q^2}{2}(1+\rho)^2$$

Since $\mathtt{ct}'_i = \mathtt{ct}_i \bmod q$ and $\|\mathtt{ct}'_i\|_\infty \leqslant \frac{q}{2}(1+\rho)$, and by using $\|\boldsymbol{s}\|_\infty \leqslant B_{key}$ we can write:

$$\mathtt{ct}'_i(\boldsymbol{s}) = \boldsymbol{c}'_{i,0} + \boldsymbol{c}'_{i,1}\cdot\boldsymbol{s} = \Delta[\boldsymbol{m}_i]_t + \boldsymbol{v}_i + q\boldsymbol{r}_i$$

with:

$$\|\boldsymbol{r}_i\|_\infty = \left\|\frac{\boldsymbol{c}'_{i,0} + \boldsymbol{c}'_{i,1}\cdot\boldsymbol{s} - \Delta[\boldsymbol{m}_i]_t - \boldsymbol{v}_i}{q}\right\|_\infty \leq (1+\rho)\frac{1+\delta_{\mathcal{R}}B_{key}}{2} + 1 = r_\infty.$$

We recall hereafter some useful bounds given in Chapter 1 which come from [BLLN13].

$$\begin{cases} \boldsymbol{v}_1\cdot\boldsymbol{v}_2 = [\boldsymbol{v}_1\boldsymbol{v}_2]_\Delta + \Delta\boldsymbol{r}_v, \ \|\boldsymbol{r}_v\|_\infty \leq \dfrac{\delta_{\mathcal{R}}}{2}\min(\|\boldsymbol{v}_1\|_\infty, \|\boldsymbol{v}_2\|_\infty) \\ [\boldsymbol{m}_1]_t\cdot[\boldsymbol{m}_2]_t = [\boldsymbol{m}_1\boldsymbol{m}_2]_t + t\boldsymbol{r}_m, \ \|\boldsymbol{r}_m\|_\infty < \dfrac{\delta_{\mathcal{R}} t}{2}. \end{cases} \tag{2.2.5}$$

By noticing that $\mathtt{ct}'_\star(\boldsymbol{s}) = (\mathtt{ct}'_1 \star \mathtt{ct}'_2)(\boldsymbol{s}) = \mathtt{ct}'_1(\boldsymbol{s})\cdot\mathtt{ct}'_2(\boldsymbol{s})$, we obtain:

$$\begin{aligned}\mathtt{ct}'_\star(\boldsymbol{s}) = & \Delta^2[\boldsymbol{m}_1]_t\cdot[\boldsymbol{m}_2]_t + \Delta([\boldsymbol{m}_1]_t\cdot\boldsymbol{v}_2 + [\boldsymbol{m}_2]_t\cdot\boldsymbol{v}_1) + q\Delta([\boldsymbol{m}_1]_t\cdot\boldsymbol{r}_2 + [\boldsymbol{m}_2]_t\cdot\boldsymbol{r}_1) \\ & + \boldsymbol{v}_1\cdot\boldsymbol{v}_2 + q(\boldsymbol{v}_1\cdot\boldsymbol{r}_2 + \boldsymbol{v}_2\cdot\boldsymbol{r}_1) + q^2\boldsymbol{r}_1\cdot\boldsymbol{r}_2.\end{aligned}$$

Then, by using (2.2.5) and $\Delta t = q - |q|_t$, we deduce that:

$$\begin{aligned}\frac{t}{q}\mathtt{ct}'_\star(\boldsymbol{s}) = & \Delta[\boldsymbol{m}_1\cdot\boldsymbol{m}_2]_t + ([\boldsymbol{m}_1]_t\cdot\boldsymbol{v}_2 + [\boldsymbol{m}_2]_t\cdot\boldsymbol{v}_1) + tq\boldsymbol{r}_1\cdot\boldsymbol{r}_2 + \boldsymbol{r}_v + \boldsymbol{r}_r \\ & + (q-|q|_t)([\boldsymbol{m}_1]_t\cdot\boldsymbol{r}_2 + [\boldsymbol{m}_2]_t\cdot\boldsymbol{r}_1 + \boldsymbol{r}_m) + t(\boldsymbol{v}_1\cdot\boldsymbol{r}_2 + \boldsymbol{v}_2\cdot\boldsymbol{r}_1).\end{aligned}$$

with $\boldsymbol{r}_r = \dfrac{t}{q}[\boldsymbol{v}\cdot\boldsymbol{v}']_\Delta - \dfrac{|q|_t}{q}(\Delta[\boldsymbol{m}_1]_t\cdot[\boldsymbol{m}_2]_t + [\boldsymbol{m}_1]_t\cdot\boldsymbol{v}_2 + [\boldsymbol{m}_2]_t\cdot\boldsymbol{v}_1 + \boldsymbol{r}_v)$ like in (1.2.18). Now

by introducing the error $\boldsymbol{r}_a$, like in (1.2.19), due to the rounding:

$$\frac{t}{q}\mathtt{ct}'_\star(\boldsymbol{s}) = \sum_{i=0}^{2} \left\lfloor \frac{t}{q}\mathtt{ct}'_\star[i] \right\rceil \cdot \boldsymbol{s}^i + \boldsymbol{r}_a.$$

We obtain:

$$\widetilde{\mathtt{ct}}_{\text{mult}}(\boldsymbol{s}) = \mathtt{DR}_{t,q}(\mathtt{ct}'_\star)(\boldsymbol{s}) = \frac{t}{q}\mathtt{ct}'_\star(\boldsymbol{s}) - \boldsymbol{r}_a = \Delta\,[\boldsymbol{m}_1\boldsymbol{m}_2]_t + \tilde{\boldsymbol{v}}_{\text{mult}} \bmod q$$

with:

$$\begin{aligned}\tilde{\boldsymbol{v}}_{\text{mult}} = {} & ([\boldsymbol{m}_1]_t \cdot \boldsymbol{v}_2 + [\boldsymbol{m}_2]_t \cdot \boldsymbol{v}_1) - |q|_t([\boldsymbol{m}_1]_t \cdot \boldsymbol{r}_2 + [\boldsymbol{m}_2]_t \cdot \boldsymbol{r}_1 + \boldsymbol{r}_m) \\ & + \boldsymbol{r}_v + t \cdot (\boldsymbol{v}_1 \cdot \boldsymbol{r}_2 + \boldsymbol{v}_2 \cdot \boldsymbol{r}_1) + \boldsymbol{r}_r - \boldsymbol{r}_a. \end{aligned} \tag{2.2.6}$$

Below we give some useful bounds:

$$\begin{cases} \|\,[\boldsymbol{m}_1]_t\,\boldsymbol{v}_2 + [\boldsymbol{m}_2]_t\,\boldsymbol{v}_1\|_\infty \leqslant \dfrac{\delta_{\mathcal{R}} t}{2}(\|\boldsymbol{v}_1\|_\infty + \|\boldsymbol{v}_2\|_\infty) \\ \|\boldsymbol{v}_1\boldsymbol{r}_2 + \boldsymbol{v}_2\boldsymbol{r}_1\|_\infty < \delta_{\mathcal{R}} r_\infty(\|\boldsymbol{v}_1\|_\infty + \|\boldsymbol{v}_2\|_\infty) \end{cases}$$

Next, by bounding each term of (2.2.6) with (1.2.6), (1.2.17), (1.2.18) and (1.2.19) and by putting all the pieces together we obtain:

$$\begin{aligned}\|\tilde{\boldsymbol{v}}_{\text{mult}}\|_\infty \leq & \delta_{\mathcal{R}} t \left(r_\infty + \frac{1}{2}\right)(\|\boldsymbol{v}_1\|_\infty + \|\boldsymbol{v}_2\|_\infty) + \frac{\delta_{\mathcal{R}}}{2}\min(\|\boldsymbol{v}_1\|_\infty, \|\boldsymbol{v}_2\|_\infty) \\ & + |q|_t \delta_{\mathcal{R}} t \left(r_\infty + \frac{3}{2}\right) + \frac{1 + \delta_{\mathcal{R}} B_{key} + \delta_{\mathcal{R}}^2 B_{key}^2}{2}\end{aligned}$$

which concludes the proof. □

2.2.3 Going back to q

Lemma 2.2.3 states that we have got back $\mathtt{DR}_2(\mathtt{ct}'_\star) + \mathtt{b}$ in $\mathcal{B}_{\mathtt{sk}}$ so far, where we have denoted $(\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2)$ by $\mathtt{b}$. To be able to perform future computations modulo q, we need to convert it back in basis q. However, the conversion has to be exact because extra multiples of $B = b_1 \dots b_\ell$ cannot be tolerated this time otherwise the noise would blow-up. The extra modulus $b_{\mathtt{sk}}$ allows us to perform a complete Shenoy and Kumaresan like conversion described in the next lemma. It is stated for a more general context where the input can be either positive or negative, and can be larger, in absolute value, than B.

Lemma 2.2.5. Let $\mathcal{B}$ be an RNS base and $b_{\mathtt{sk}}$ be a modulus coprime to $B = \prod_{b\in\mathcal{B}} b$. Let x be an integer such that $|x| < \lambda B$ (for some real number $\lambda \geqslant 1$) and whose residues are given

in $\mathcal{B}_{\mathtt{sk}}$. Let's assume that $b_{\mathtt{sk}}$ satisfies $b_{\mathtt{sk}} \geqslant 2(|\mathcal{B}| + \lceil \lambda \rceil)$. Let α_x be the following integer:

$$\alpha_x := \left[(\mathtt{FastBconv}(x, \mathcal{B}, \{b_{\mathtt{sk}}\}) - x_{\mathtt{sk}}) B^{-1} \right]_{b_{\mathtt{sk}}}. \tag{2.2.7}$$

Then, for x being either positive or negative, the following equality holds:

$$\mathtt{FastBconvSK}(x, \mathcal{B}_{\mathtt{sk}}, q) := (\mathtt{FastBconv}(x, \mathcal{B}, q) - \alpha_x B) \bmod q = x \bmod q. \tag{2.2.8}$$

Proof. We set $\mathcal{B} = \{b_1, \ldots, b_\ell\}$ ($|\mathcal{B}| = \ell$), the possible negativity of x is the reason why we have to compute a centered remainder modulo m_{sk} in (2.2.7). First, we notice that the residues of x in $\mathcal{B}$ are actually those of $|x|_B = x + \mu B \in [0, B)$, where μ is an integer lying in $[-\lambda + 1, \lambda] \cap \mathbb{Z}$. Therefore, we deduce that:

$$\sum_{i=1}^{\ell} \left| x \frac{b_i}{B} \right|_{b_i} \frac{B}{b_i} = |x|_B + \alpha_{|x|_B} B = (x + \mu B) + \alpha_{x+\mu B} B$$

with $0 \leqslant \alpha_{x+\mu M} \leqslant \ell - 1$. Denoting $|x|_{b_{\mathtt{sk}}}$ by $x_{\mathtt{sk}}$, it follows that the quantity computed in (2.2.7) is the following one:

$$\begin{aligned} \left[\left(\sum_{i=1}^{\ell} \left| x \frac{b_i}{B} \right|_{b_i} \frac{B}{b_i} - x_{\mathtt{sk}} \right) B^{-1} \right]_{b_{\mathtt{sk}}} &= \left[\left(\sum_{i=1}^{\ell} \left| x \frac{b_i}{B} \right|_{b_i} \frac{B}{b_i} - (x_{\mathtt{sk}} + \mu B) \right) B^{-1} + \mu \right]_{b_{\mathtt{sk}}} \\ &= \left[\alpha_{x+\mu B} + \mu \right]_{b_{\mathtt{sk}}}. \end{aligned}$$

It remains to show that $[\alpha_{(x+\mu B)} + \mu]_{b_{\mathtt{sk}}} = \alpha_{x+\mu B} + \mu$ or, in other words, that:

$$-\left\lfloor \frac{b_{sk}}{2} \right\rfloor \leqslant \alpha_{x+\mu B} + \mu \leqslant \left\lfloor \frac{b_{sk} - 1}{2} \right\rfloor$$

But by hypothesis on $b_{\mathtt{sk}}$, and because $\ell \geqslant 1$, we can write $b_{\mathtt{sk}} \geqslant 2(\ell + \lceil \lambda \rceil) \geqslant 2\lfloor \lambda \rfloor + 1$. Then,

$$\begin{cases} \alpha_{x+\mu B} + \mu &\leqslant \ell - 1 + \lceil \lambda \rceil \leqslant \dfrac{b_{sk}}{2} - 1 \leqslant \left\lfloor \dfrac{b_{sk} - 1}{2} \right\rfloor, \\ \alpha_{x+\mu M} + \mu &\geqslant -\lfloor \lambda \rfloor \geqslant -\dfrac{b_{sk} - 1}{2} \geqslant -\left\lfloor \dfrac{b_{sk}}{2} \right\rfloor. \end{cases}$$

Thus, $[\alpha_{x+\mu B} + \mu]_{b_{\mathtt{sk}}} = \alpha_{x+\mu B} + \mu$, and it follows that, by computing the right member of (2.2.8), we obtain what we wanted:

$$\sum_{i=1}^{\ell} \left| x \frac{b_i}{B} \right|_{b_i} \frac{B}{b_i} - [\alpha_{x+\mu B} + \mu]_{b_{\mathtt{sk}}} B = (x + \mu B) + \alpha_{x+\mu B} B - (\alpha_{x+\mu B} + \mu) B = x.$$

□

Therefore we can convert our ciphertext back to q through the following proposition:

Proposition 2.2.6. Given a positive real number λ, let $b_{\mathtt{sk}}$ and $\mathcal{B}$ be such that:

$$\lambda B > \delta_{\mathcal{R}} t \frac{q}{2}(1+\rho)^2 + \tfrac{1}{2} + k \text{ and } \; b_{\mathtt{sk}} \geqslant 2(|\mathcal{B}| + \lceil \lambda \rceil). \tag{2.2.9}$$

Let's assume that $\mathtt{DR}_2(\mathtt{ct}'_\star) + \mathtt{b}$ is given in $\mathcal{B}_{\mathtt{sk}}$, with $\|\mathtt{b}\|_\infty \leqslant k$. Then,

$$\mathtt{FastBconvSK}(\mathtt{DR}_2(\mathtt{ct}'_\star) + \mathtt{b}, \mathcal{B}_{\mathtt{sk}}, q) = \left(\mathtt{DR}_2(\mathtt{ct}'_\star) + \mathtt{b}\right) \bmod q.$$

Proof. By using (2.2.2), we have:

$$\|\mathtt{DR}_2(\mathtt{ct}'_\star)\|_\infty \leqslant \left\| \frac{t}{q} \mathtt{ct}'_\star \right\|_\infty + \frac{1}{2} \leqslant \delta_{\mathcal{R}} t \frac{q}{2}(1+\rho)^2 + \frac{1}{2}$$

Lemma 2.2.5 concludes the proof. □

Algorithm 16 summarizes the new full RNS variant $\mathtt{Mult}_{\mathtt{RNS}}$.

Algorithm 16 RNS homomorphic multiplication $\mathtt{Mult}_{\mathtt{RNS}}$

Require: $\mathtt{ct}_1, \mathtt{ct}_2$ in q with Montgomery representation with respect to $\tilde{b}$
Ensure: $\mathtt{ct}_{\text{mult}}$ in q with Montgomery representation with respect to $\tilde{b}$
S0: Convert fast $\mathtt{ct}_1$ and $\mathtt{ct}_2$ from q to $\mathcal{B}_{\mathtt{sk}} \cup \{\tilde{b}\}$: $\rightsquigarrow \mathtt{ct}''_i = \mathtt{ct}_i + q$-overflows
S1: Reduce q-overflows in $\mathcal{B}_{\mathtt{sk}}$: $\rightsquigarrow \mathtt{ct}'_i$ (in $\mathcal{B}_{\mathtt{sk}}$) $\leftarrow \mathtt{SmMRq}_{\tilde{m}}(((\mathtt{ct}''_i)_b)_{b \in \mathcal{B}_{\mathtt{sk}} \cup \{\tilde{b}\}})$
S2: Compute the product $\mathtt{ct}'_\star = \mathtt{ct}'_1 \star \mathtt{ct}'_2$ in $q \cup \mathcal{B}_{\mathtt{sk}}$
S3: Convert fast from q to $\mathcal{B}_{\mathtt{sk}}$ to achieve first step (approximate rounding) in $\mathcal{B}_{\mathtt{sk}}$:
$(\widetilde{\mathtt{ct}}_{\text{mult}} + \mathtt{b} = \mathtt{DR}_{t,q}(\mathtt{ct}'_\star) + \mathtt{b}$ in $\mathcal{B}_{\mathtt{sk}}) \leftarrow \ldots \leftarrow \mathtt{FastBconv}(t \cdot \mathtt{ct}'_\star, q, \mathcal{B}_{\mathtt{sk}})$
S4: Convert exactly from $\mathcal{B}_{\mathtt{sk}}$ to q to achieve transitional step:
$(\widetilde{\mathtt{ct}}_{\text{mult}} + \mathtt{b}$ in $q) \leftarrow \mathtt{FastBconvSK}(\widetilde{\mathtt{ct}}_{\text{mult}} + \mathtt{b}, \mathcal{B}_{\mathtt{sk}}, q)$

2.3 Last step: the relinearization

At this point, $\widetilde{\mathtt{ct}}_{\text{mult}} + \mathtt{b} = (\overline{c}_0, \overline{c}_1, \overline{c}_2)$ is known in base q ($\widetilde{\mathtt{ct}}_{mult} := \mathtt{DR}_2(\mathtt{ct}'_\star)$). The main issue with the original procedure is that the function $\mathcal{D}_{\omega,q}$ (in $\mathtt{Relin}_{\mathtt{FV}}$) requires, by definition, an access to a positional system (in radix base ω), which is hardly compatible with RNS

2.3.1 Adapting the original procedure

We recall that the original relinearization procedure is done as follows:

$$\mathtt{ct}_{\text{mult}} = \left([\overline{c}_0 + \langle \mathcal{D}_{\omega,q}(\overline{c}_2), \mathcal{P}_{\omega,q}(s^2) - (\vec{e} + s\vec{a}) \rangle]_q, [\overline{c}_1 + \langle \mathcal{D}_{\omega,q}(\overline{c}_2), \vec{a} \rangle]_q \right) \tag{2.3.1}$$

where $\vec{e} \leftarrow \chi_{err}^{\ell_{\omega,q}}$, $\vec{a} \leftarrow \mathcal{U}_q^{\ell_{\omega,q}}$. The decomposition of $\overline{c}_2$ in radix ω enables a crucial reduction of the noise growth due to the multiplications by the terms $e_i + (s \cdot a)_i$. It cannot be done directly in RNS as is. Indeed, it would require a costly switch between RNS and positional representation in radix ω. However, we can do something very similar. We recall that we can

write:

$$\overline{c}_2 = \sum_{i=1}^{k} \left| \overline{c}_2 \frac{q_i}{q} \right|_{q_i} \times \frac{q}{q_i} \bmod q$$

If ω has the same order of magnitude as 2^ν (size of moduli in q), we obtain a similar limitation of the noise growth by using the vectors:

$$\mathcal{D}_{\mathtt{RNS},q}(\overline{c}_2) = \left(\left| \overline{c}_2 \frac{q_1}{q} \right|_{q_1}, \ldots, \left| \overline{c}_2 \frac{q_k}{q} \right|_{q_k} \right) \text{ and } \mathcal{P}_{\mathtt{RNS},q}(s^2) = \left(\left| s^2 \frac{q}{q_1} \right|_{q}, \ldots, \left| s^2 \frac{q}{q_k} \right|_{q} \right)$$

both in $\mathcal{R}^k$. This decomposition has the same properties as the original one:

Lemma 2.3.1. For any $c \in \mathcal{R}$, we have $\left\langle \mathcal{D}_{\mathtt{RNS},q}(c), \mathcal{P}_{\mathtt{RNS},q}(s^2) \right\rangle \equiv cs^2 \bmod q$.

Thus, by replacing the public relinearization key $\overrightarrow{\mathtt{rlk}_{\mathtt{FV}}}$ by:

$$\overrightarrow{\mathtt{rlk}}_{\mathtt{RNS}} = \left([\mathcal{P}_{\mathtt{RNS},q}(s^2) + (\vec{e} + s \cdot \vec{a})]_q, -\vec{a} \right)$$

with $\vec{a} \xleftarrow{\$} \mathcal{U}_q^k$ and $\vec{e} \xleftarrow{\$} \chi_{err}^k$ and which can be seen as the matrices:

$$\begin{aligned} \overrightarrow{\mathtt{rlk}}_{\mathtt{RNS}}[0] &= \begin{pmatrix} \left| s^2 \frac{q}{q_1} + e_1 + s \cdot a_1 \right|_{q_1} & \cdots & |e_1 + s \cdot a_1|_{q_k} \\ \vdots & & \vdots \\ |e_k + s \cdot a_k|_{q_1} & \cdots & \left| s^2 \frac{q}{q_k} + e_k + s \cdot a_k \right|_{q_k} \end{pmatrix} \\ \overrightarrow{\mathtt{rlk}}_{\mathtt{RNS}}[1] &= \begin{pmatrix} |-a_1|_{q_1} & \cdots & |-a_1|_{q_k} \\ \vdots & & \vdots \\ |-a_k|_{q_1} & \cdots & |-a_k|_{q_k} \end{pmatrix} \end{aligned} \tag{2.3.2}$$

we can perform the relinearization step in $\mathtt{RNS}$. The next lemma helps for providing a bound on the extra noise introduced by this step.

Lemma 2.3.2. Let $\vec{e} \leftarrow \chi_{err}^k$, $\vec{a} \leftarrow \mathcal{U}_q^k$, and $c \in R$. Then,

$$\left\| \left(\left\langle \mathcal{D}_{\mathtt{RNS},q}(c), (\vec{e} + \vec{a}s) \right\rangle + \left\langle \mathcal{D}_{\mathtt{RNS},q}(c), \overrightarrow{-a} \right\rangle s \right) \bmod q \right\|_\infty < \delta_{\mathcal{R}} B_{err} k 2^\nu. \tag{2.3.3}$$

Proof. First, we have:

$$\left(\left\langle \mathcal{D}_{\mathtt{RNS},q}(c), +(\vec{e} + \vec{a}s) \right\rangle + \left\langle \mathcal{D}_{\mathtt{RNS},q}(c), -\vec{a} \right\rangle s \right) = \left\langle \mathcal{D}_{\mathtt{RNS},q}(c), \vec{e} \right\rangle \bmod q$$

Second, by using $q_i < 2^\nu$, we obtain:

$$\left\| \left\langle \mathcal{D}_{\mathtt{RNS},q}(c), \vec{e} \right\rangle \right\|_\infty = \left\| \sum_{i=1}^{k} \left| c \frac{q_i}{q} \right|_{q_i} \cdot e_i \right\|_\infty < \sum_{i=1}^{k} \delta_{\mathcal{R}} q_i \|e_i\|_\infty \leqslant \delta_{\mathcal{R}} B_{err} \sum_{i=1}^{k} q_i < \delta_{\mathcal{R}} B_{err} k 2^\nu$$

□

Remark 2.3.3. It is still possible to add a second level of decomposition (like in original

approach, but applied on the residues) to limit a little bit more the noise growth (section 2.3.2). Furthermore, Sect. 2.3.3 details how the size of $\texttt{rlk}_{\texttt{RNS}}$ can be reduced in a similar way that $\texttt{rlk}_{\texttt{FV}}$ could be through the method described in ([BLLN13], 5.4).

Finally, on input $\overline{\texttt{ct}} = \widetilde{\texttt{ct}}_{\text{mult}} + \texttt{b} = (\overline{c}_0, \overline{c}_1, \overline{c}_2)$, and the relinearization key $\texttt{rlk}_{\texttt{RNS}} = \left([\mathcal{P}_{\texttt{RNS},q}(s^2) + (\vec{e} + s\vec{a})]_q, -\vec{a}\right)$, the output of the RNS relinearization is the following one:

$$\texttt{ct}_{\text{mult}} = \left(\left[\overline{c}_0 + \left\langle \mathcal{D}_{\texttt{RNS},q}(\overline{c}_2), \mathcal{P}_{\texttt{RNS},q}(s^2) + (\vec{e} + \vec{a}\,s)\right\rangle\right]_q, \left[\overline{c}_1 + \left\langle \mathcal{D}_{\texttt{RNS},q}(\overline{c}_2), \overline{-\vec{a}}\right\rangle\right]_q\right) \tag{2.3.4}$$

Proposition 2.3.4. Let $\texttt{ct}_{\text{mult}}$ be as in (2.3.4), and v_{mult} (resp. $\widetilde{v}_{\text{mult}}$) the inherent noise of $\texttt{ct}_{\text{mult}}$ (resp. $\widetilde{\texttt{ct}}_{\text{mult}}$). Then:

$$\texttt{ct}_{\text{mult}}(s) = \Delta\,[m_1 m_2]_t + v_{\text{mult}} (\text{mod } q)$$

with:

$$\|v_{\text{mult}}\|_\infty < \|\widetilde{v}_{\text{mult}}\|_\infty + k(1 + \delta_{\mathcal{R}} B_{key}(1 + \delta_{\mathcal{R}} B_{key})) + \delta_{\mathcal{R}} k B_{err} 2^{\nu+1}. \tag{2.3.5}$$

Proof. At this point, we are evaluating $\texttt{Relin}_{\texttt{RNS}}(\widetilde{\texttt{ct}}_{\text{mult}} + \texttt{b})$. We denote $\widetilde{\texttt{ct}}_{\text{mult}} = (c_0, c_1, c_2)$ and $\texttt{b} = (b_0, b_1, b_2)$. We can first notice that:

$$\mathcal{D}_{\texttt{RNS},q}(c_2 + b_2) = \mathcal{D}_{\texttt{RNS},q}(c_2) + \mathcal{D}_{\texttt{RNS},q}(b_2) - \vec{u}$$

where $\vec{u} = (u_1, \ldots, u_k) \in \mathcal{R}^k$ is such that, for any $(i, j) \in [1, k] \times [0, n-1]$, $(u_i)_j \in \{0, q_i\}$. In particular, it can be noticed that $\| |b_2 \frac{q_i}{q}|_{q_i} - u_i \|_\infty < q_i$, and that $\left\langle \vec{u}, \mathcal{P}_{\texttt{RNS},q}(s^2) \right\rangle \equiv 0 \bmod q$. This latter fact leads to the following equality:

$$\begin{aligned}\left\langle \vec{u}, \texttt{rlk}_{\texttt{RNS}}[0] \right\rangle + \left\langle \vec{u}, \texttt{rlk}_{\texttt{RNS}}[1] \right\rangle \cdot s &= \left\langle \vec{u}, \mathcal{P}_{\texttt{RNS},q}(s^2) + (\vec{e} + s\vec{a}) \right\rangle + \left\langle \vec{u}, -\vec{a} \right\rangle \cdot s \\ &= \left\langle \vec{u}, \vec{e} \right\rangle = \sum_{i=1}^{k} u_i \cdot e_i \mod q\end{aligned}$$

Consequently, in $\mathcal{R}_q \times \mathcal{R}_q$ we have that:

$$\begin{aligned}\texttt{Relin}_{\texttt{RNS}}(\overline{\texttt{ct}}) &= \left(\left[\overline{c}_0 + \left\langle \mathcal{D}_{\texttt{RNS},q}(\overline{c}_2), \mathcal{P}_{\texttt{RNS},q}(s^2) + (\vec{e} + \vec{a}\,s)\right\rangle\right]_q, \left[\overline{c}_1 + \left\langle \mathcal{D}_{\texttt{RNS},q}(\overline{c}_2), \overline{-\vec{a}}\right\rangle\right]_q\right) \\ &= \texttt{Relin}_{\texttt{RNS}}(\widetilde{\texttt{ct}}_{\text{mult}}) + \texttt{Relin}_{\texttt{RNS}}(\texttt{b}) - \left(\left\langle \vec{u}, \texttt{rlk}_{\texttt{RNS}}[0] \right\rangle, \left\langle \vec{u}, \texttt{rlk}_{\texttt{RNS}}[1] \right\rangle\right)\end{aligned}$$

Thus, a part of the noise comes from the following extra term:

$$\begin{aligned}
&\left\|\mathtt{Relin}_{\mathtt{RNS}}(\mathtt{b})(\boldsymbol{s}) - \left\langle \vec{\boldsymbol{u}}, \mathtt{rlk}_{\mathtt{RNS}}[0]\right\rangle - \left\langle \vec{\boldsymbol{u}}, \mathtt{rlk}_{\mathtt{RNS}}[1]\right\rangle \cdot \boldsymbol{s}\right\|_\infty \\
=&\left\|\boldsymbol{b}_0 + \left\langle \mathcal{D}_{\mathtt{RNS},q}(\boldsymbol{b}_2) - \vec{\boldsymbol{u}}, \mathtt{rlk}_{\mathtt{RNS}}[0]\right\rangle + \boldsymbol{s}\cdot(\boldsymbol{b}_1 + \left\langle \mathcal{D}_{\mathtt{RNS},q}(\boldsymbol{b}_2) - \vec{\boldsymbol{u}}, \mathtt{rlk}_{\mathtt{RNS}}[1]\right\rangle\right\|_\infty \\
=&\left\|\boldsymbol{b}_0 + \boldsymbol{b}_1\cdot\boldsymbol{s} + \boldsymbol{b}_2\cdot\boldsymbol{s}^2 - \sum_{i=1}^{k}\left(\left|\boldsymbol{b}_2\frac{q_i}{q}\right|_{q_i} - \boldsymbol{u}_i\right)\cdot\boldsymbol{e}_i\right\|_\infty \\
<&k(1+\delta_{\mathcal{R}}B_{key} + \delta_{\mathcal{R}}^2 B_{key}^2) + \delta_{\mathcal{R}}B_{err}\sum_{i=1}^{k} q_i \\
<&k(1+\delta_{\mathcal{R}}B_{key} + \delta_{\mathcal{R}}^2 B_{key}^2) + \delta_{\mathcal{R}}B_{err}k2^{\nu}.
\end{aligned}$$

and lemma 2.3.2 brings the rest of the noise. □

2.3.2 Combining two levels of decomposition

To reduce the noise growth due to the relinearization step a bit more, we can integrate another level of decomposition in radix ω where $\omega = 2^{\theta} << 2^{\nu}$ as efficiently as in the original scheme by doing it on the residues, because they are handled through the classical binary positional system. By denoting $\ell_{\omega,2^{\nu}} = \left\lceil \frac{\nu}{\theta}\right\rceil$, each polynomial $\left|\boldsymbol{c}\frac{q_i}{q}\right|_{q_i}$ is decomposed into the vector of polynomials $\left(\left[\left\lfloor |\boldsymbol{c}\frac{q_i}{q}|_{q_i}\omega^{-z}\right\rfloor\right]_{\omega}\right)_{z\in[0,\ldots,\ell_{\omega,2^{\nu}}-1]}$, and the new decomposition function is defined by:

$$\mathcal{D}_{\mathtt{RNS},\omega,q}(\boldsymbol{c}) = \left(\boldsymbol{d}_i^z = \left[\left\lfloor\left|\boldsymbol{c}\frac{q_i}{q}\right|_{q_i}\omega^{-z}\right\rfloor\right]_w\right)_{i\in[1,k],z\in[0,\ldots,\ell_{\omega,2^{\nu}}-1]}.$$

Therefore, each term $\left|\boldsymbol{s}^2\frac{q}{q_i} + (\boldsymbol{e}_i + (\boldsymbol{s}\cdot\boldsymbol{a})_i)\right|_{q_j}$ in $\mathtt{rlk}_{\mathtt{RNS}}[0]$ has to be replaced by:

$$\left(\left|\boldsymbol{s}^2\frac{q}{q_i}\omega^z + (\boldsymbol{e}_i^z + (\boldsymbol{s}\cdot\boldsymbol{a})_i^z)\right|_{q_j}\right)_z,\ z = 0,\ldots,\ell_{\omega,2^{\nu}}-1,\ \boldsymbol{e}_i^z \leftarrow \chi_{err}, \boldsymbol{a}_i^z \leftarrow \mathcal{U}_q.$$

It follows that the extra noise is now bounded by:

$$\left\|\left\langle \mathcal{D}_{\mathtt{RNS},\omega,q}(\boldsymbol{c}), \vec{\boldsymbol{e}}\right\rangle\right\|_\infty = \left\|\sum_{i=1}^{k}\sum_{z=0}^{\ell_{\omega,2^{\nu}}-1}\boldsymbol{d}_i^z\boldsymbol{e}_i^z\right\|_\infty < \delta_{\mathcal{R}}B_{err}\omega k\ell_{\omega,2^{\nu}}.$$

In other words, the term 2^{ν} in (2.3.3) is replaced by $\omega\ell_{\omega,2^{\nu}}$. Even if such optimization may be not really relevant in practice, since the noise caused by the relinearization step is very small compared to the one caused by the multiplication. This is another argument for saying that any trick and technique of the original scheme can be efficiently adapted to a pure RNS approach.

2.3.3 Reducing the size of the relinearization key $\mathtt{rlk}_{\mathtt{RNS}}$

In [BLLN13], Sect. 5.4, the authors describe a method to significantly reduce the size of the public relinearization key `rlk` (by truncating the ciphertext) which can be applied to the

original FV scheme. We provide an efficient adaptation of such kind of optimization to the RNS variant of the relinearization step.

We recall that the relinearization is applied to a degree-2 ciphertext denoted here by $(\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2)$. The initial suggestion was to set to zero, say, the i lowest significant components of the vector $\mathcal{D}_{\omega,q}(\boldsymbol{c}_2)$. Doing so is equivalent to replacing $\boldsymbol{c}_2$ by $\boldsymbol{c}_2' = \omega^i \lfloor \boldsymbol{c}_2 \omega^{-i} \rfloor = \boldsymbol{c}_2 - |\boldsymbol{c}_2|_{\omega^i}$. Thus, only the $\ell_{\omega,q} - i$ most significant components of $\texttt{rlk}_{\texttt{FV}}[0]$ (and then of $\texttt{rlk}_{\texttt{FV}}[1]$) are required (in other words, when $\texttt{rlk}_{\texttt{FV}}[0]$ is viewed as an $(\ell_{q,\omega} \times k)$ RNS matrix by decomposing each component in base q, ik entries are set to zero like this). However this optimization causes a greater noise than the one in lemma 4 of [BLLN13], indeed given $(\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2)$ decryptable under $\boldsymbol{s}$, the relinearization step provides:

$$(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1) := \left(\boldsymbol{c}_0 + \left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_2'), \mathcal{P}_{\omega,q}(\boldsymbol{s}^2) + (\vec{\boldsymbol{e}} + \vec{\boldsymbol{a}} \cdot \boldsymbol{s}) \right\rangle, \boldsymbol{c}_1 + \left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_2'), -\vec{\boldsymbol{a}} \right\rangle\right)$$

Thus, $(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1)(\boldsymbol{s}) = \boldsymbol{c}_0 + \boldsymbol{c}_1\boldsymbol{s} + \boldsymbol{c}_2'\boldsymbol{s}^2 + \left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_2'), \vec{\boldsymbol{e}} \right\rangle \bmod q$. Consequently, the extra noise comes from the following term:

$$\begin{aligned} \left\| -|\boldsymbol{c}_2|_{\omega^i} \boldsymbol{s}^2 + \left\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}_2'), \vec{\boldsymbol{e}} \right\rangle \right\|_\infty &= \left\| -|\boldsymbol{c}_2|_{\omega^i} \cdot \boldsymbol{s}^2 + \sum_{j=i}^{\ell_{\omega,q}-1} \mathcal{D}_{\omega,q}(\boldsymbol{c}_2)_j \cdot \boldsymbol{e}_j \right\|_\infty \\ &< \delta_{\mathcal{R}}^2 \omega^i B_{key}^2 + (\ell_{\omega,q} - i)\delta_{\mathcal{R}} \omega B_{err}. \end{aligned} \tag{2.3.6}$$

In the present RNS variant, the computation of $\lfloor \boldsymbol{c}_2 \omega^{-i} \rfloor$ is not straightforward. This could be replaced by $\lfloor \boldsymbol{c}_2 (q_1 \dots q_i)^{-1} \rfloor$ through a Newton like interpolation (also known as mixed-radix conversion [ST67]). Though the result would be quite similar to the original optimization in terms of noise growth, its efficiency is not satisfying. Indeed, despite ik entries of the RNS matrix $\texttt{rlk}_{\texttt{RNS}}[0]$ can be set to zero like this, this Newton interpolation is intrinsically sequential, while the division by ω^i is just an immediate zeroing of the lowest significant coefficients in radix ω representation. Furthermore, a direct approach consisting in zeroing, say, the first i components of $\mathcal{D}_{\texttt{RNS},q}(\boldsymbol{c}_2)$ cannot work. Indeed, it would be like using $\mathcal{D}_{\texttt{RNS},q}(Q_i \times |\boldsymbol{c}_2 Q_i^{-1}|_{q_{i+1} \dots q_k})$ with $Q_i = q_1 \cdots q_i$. Then, when evaluating the output of relinearization in the secret key $\boldsymbol{s}$, it would introduce the term:

$$\begin{aligned} \left\langle \mathcal{D}_{\texttt{RNS},q}\left(Q_i \times |\boldsymbol{c}_2 Q_i^{-1}|_{q_{i+1} \cdots q_k}\right), \mathcal{P}_{\texttt{RNS},q}(\boldsymbol{s}^2) \right\rangle &= Q_i \times \left|\boldsymbol{c}_2 Q_i^{-1}\right|_{q_{i+1} \cdots q_k} \boldsymbol{s}^2 \bmod q \\ &= \left(\boldsymbol{c}_2 - q_{i+1} \cdots q_k \times \left|\boldsymbol{c}_2 (q_{i+1} \cdots q_k)^{-1}\right|_{Q_i}\right) \boldsymbol{s}^2 \bmod q \end{aligned}$$

with the norm of $\left|\boldsymbol{c}_2 (q_{i+1} \cdots q_k)^{-1}\right|_{Q_i}$ which has no reason to be small.

For our approach, we rely on the fact that $\texttt{rlk}_{\texttt{RNS}}$ contains Ring-LWE encryptions of the polynomials $\left|\boldsymbol{s}^2 \frac{q}{q_j}\right|_q$. Notice that only the j^{th}-residue of $\left|\boldsymbol{s}^2 \frac{q}{q_j}\right|_q$ can be non zero (cf. Equation 2.3.2). So, let's assume that we want to cancel ik entries in $\texttt{rlk}_{\texttt{RNS}}[0]$ (as it has been done in $\texttt{rlk}_{\texttt{FV}}$ with the previous optimization). Then we choose, for each index j, a subset of index-

numbers $\mathcal{I}_j \subseteq [1,k] \setminus \{j\}$ with cardinality i of zero elements (i.e. at line j of $\texttt{rlk}_{\texttt{RNS}}$, choose i elements, except the diagonal one , and set them to zero).

Next, for each j, we introduce a Ring-LWE-encryption of $\left|s^2 \frac{q}{q_j q_{\mathcal{I}_j}}\right|_q$, where $q_{\mathcal{I}_j} = \prod_{s\in\mathcal{I}_j} q_s$, which is $\left(\left|s^2 \frac{q}{q_j q_{\mathcal{I}_j}} + (e_j + s\cdot a_j)\right|_q, -a_j\right)$. So far, the underlying security features are still relevant. Now, it remains to multiply this encryption by $q_{\mathcal{I}_j}$, which gives $\left|s^2 \frac{q}{q_j} + q_{\mathcal{I}_j}(e_j + s\cdot a_j)\right|_q$. This corresponds to the j^{th}-line of the new matrix $\texttt{rlk}'_{\texttt{RNS}}[0]$. It is clear that this line contains zeros at columns index-numbered by $\mathcal{I}_j$. $\texttt{rlk}_{\texttt{RNS}}[1] = (a_1,\dots,a_k)$ is replaced by $\texttt{rlk}'_{\texttt{RNS}}[1] = (|q_{\mathcal{I}_1}a_1|_q,\dots,|q_{\mathcal{I}_k}a_k|_q)$.

Let's analyze the noise growth caused by the relinearization with this new relinearization key. By evaluating in s the output of relinearization with this new $\texttt{rlk}'_{\texttt{RNS}}$, we obtain:

$$\begin{aligned}
& c_0 + \left\langle \mathcal{D}_{\texttt{RNS},q}(c_2), \texttt{rlk}'_{\texttt{RNS}}[0]\right\rangle + \left(c_1 + \left\langle \mathcal{D}_{\texttt{RNS},q}(c_2), \texttt{rlk}'_{\texttt{RNS}}[1]\right\rangle\right)\cdot s \\
= \;& c_0 + \sum_{j=1}^{k} \left|c_2 \frac{q_j}{q}\right|_{q_j} \left(s^2 \frac{q}{q_j} + q_{\mathcal{I}_j}(e_j + s\cdot a_j)\right) + \left(c_1 - \sum_{j=1}^{k} \left|c_2 \frac{q_j}{q}\right|_{q_j} q_{\mathcal{I}_j} a_j\right)\cdot s \\
= \;& c_0 + c_1\cdot s + c_2\cdot s^2 - \sum_{j=1}^{k} \left|c_2 \frac{q_j}{q}\right|_{q_j} \cdot q_{\mathcal{I}_j} e_j
\end{aligned}$$

Consequently, the cancellation of ik terms in the public matrix $\texttt{rlk}_{\texttt{RNS}}[0]$ by using this method causes an extra noise growth bounded by (this can be fairly compared to (2.3.6) in the case where $\omega = 2^\nu$, i.e. $k = \ell_{\omega,q}$):

$$\left\| \sum_{j=1}^{k} \left|c_2 \tfrac{q_j}{q}\right|_{q_j} \cdot q_{\mathcal{I}_j} e_j \right\|_\infty < \sum_{j=1}^{k} \delta_{\mathcal{R}} q_j q_{\mathcal{I}_j} B_{err} < \delta_{\mathcal{R}} k 2^{\nu(i+1)} B_{err}$$

Therefore, the truncation of ciphertexts can be efficiently adapted to RNS representation without causing more significant noise growth.

2.3.4 About computational complexity

We analyze the cost of a multi-precision variant, in order to estimate the benefits of the new RNS variant of multiplication in terms of computational cost.

- The product $\texttt{ct}_\star = \texttt{ct}_1 \star \texttt{ct}_2 = (c_{1,0}, c_{1,1}) \star (c_{2,0}, c_{2,1})$ in MP variant is advantageously performed in RNS, in order to benefit from NTT. So, the MP variant considered here is assumed to use a second base $\mathcal{B}'$ such that $qB' > \|\texttt{ct}_\star\|_\infty$. By taking centered remainders modulo q, we consider $\|\texttt{ct}_i\|_\infty \leqslant \frac{q}{2}$. Then $\mathcal{B}'$ must verify in particular that:

  $$\|\texttt{ct}_\star[1]\|_\infty = \|c_{1,0}c_{2,1} + c_{1,1}c_{2,0}\|_\infty \leqslant 2\delta_{\mathcal{R}} \frac{q^2}{4} < qB'$$

  Thus, $|\mathcal{B}'|$ has to be at least equal to $k+1$ (notice that, in RNS variant, we also need to have $|\mathcal{B}_{sk}| = k+1$ for dealing with the multiplication).

- The conversion, from q to $\mathcal{B}'$, of each $\mathtt{ct}_i$ has to be as exact as possible in order to reduce the noise growth. It can be done by computing: $[\sum_{i=1}^{k} |c_{a,b}|_{q_i} \times (|\frac{q_i}{q}|_{q_i} \frac{q}{q_i})]_q$ in $\mathcal{B}'$. The terms $(|\frac{q_i}{q}|_{q_i} \frac{q}{q_i})$ are precomputable and their size is k words ($\log_2(q)$ bits). Thus, the sum involves $k^2 n\mathtt{EM}$. The reduction modulo q can be performed by using an efficient reduction as described in [AMBG+16], reducing to around 2 multiplications of k-word integers, that is $O(k^{1+\varepsilon})n\mathtt{EM}$ (where ε stands for complexity of multi-precision multiplication in radix-base 2^ν; e.g. $\varepsilon = 1$ for the schoolbook multiplication). Next, the k-word value is reduced modulo each 1-word element of $\mathcal{B}'$, through around $2kn\mathtt{EM}$ for the whole set of coefficients. Finally, this procedure has to be executed four times. Its total cost is around $(4k^2 + O(k^{1+\varepsilon}))n\mathtt{EM}$.

- Next, the product $\mathtt{ct}_1 \star \mathtt{ct}_2$ is done in $q \cup \mathcal{B}'$. First, $4(2k+1)\mathtt{NTT}$ are applied. Second, by using a Karatsuba like trick, the product is achieved by using only $3\times(2k+1)n\mathtt{IMM}$. Third, $3(2k+1)\mathtt{invNTT}$ are applied to recover $\mathtt{ct}_\star = (c_{\star,0}, c_{\star,1}, c_{\star,2})$ in coefficient representation.

- The next step is the division and rounding of the three polynomials $c_{\star,i}$'s. A lift from $q \cup \mathcal{B}'$ to $\mathbb{Z}$ is required, for a cost of $3(2k+1)^2 n\mathtt{EM}$. t/q can be precomputed with around $3k + 1$ words of precision to ensure a correct rounding. Thus, a product $\frac{t}{q} \times c_{\star,i}$ is achieved with $O(k^{1+\varepsilon})n\mathtt{EM}$. After, the rounding of $c_{\star,0}$ and $c_{\star,1}$ are reduced in RNS base q by $2 \times 2kn\mathtt{EM}$.

- The relinearization step (2.3.1) can be done in each RNS channel of q. By assuming that $\omega = 2^\nu$, we would have $\ell_{\omega,q} = k$. The computation of the vector $\mathcal{D}_{\omega,q}(\lfloor \frac{t}{q} c_2 \rceil)$ reduces to shifting. The two scalar products in $\mathtt{Relin}_{\mathtt{FV}}$, with an output in coefficient representation, require $k\ell_{\omega,q}\mathtt{NTT} + 2k^2 n\mathtt{IMM} + 2k\mathtt{invNTT}$.

Thus, the total cost of the multi-precision variant is at most the following one:

$$\mathtt{Cost}(\mathtt{Mult}_{\mathtt{MP}}) = (k\ell_{\omega,q} + 8k + 4)\mathtt{NTT} + (8k+3)\mathtt{invNTT} + [2k^2 + 6k + 3]n\mathtt{IMM} + [40k^2 + O(k^{1+\varepsilon})]n\mathtt{EM}.$$

Now, let's analyze the cost of the RNS variant.

- The fast conversions from q to $\mathcal{B}_{sk} \cup \{\tilde{m}\}$ require $4 \times k(k+2)n\mathtt{IMM}$.

- The reduction of q-overflows requires $4 \times (k+1)n\mathtt{IMM}$.

- The product of ciphertexts $\mathtt{ct}'_1 \star \mathtt{ct}'_2$ in $q \cup \mathcal{B}_{sk}$ requires the same cost as for MP variant, that is $4(2k+1)\mathtt{NTT} + 3(2k+1)n\mathtt{IMM} + 3(2k+1)\mathtt{invNTT}$.

- With adequate pre-computed data, the base conversion from q to $\mathcal{B}_{\mathtt{sk}}$ can integrate the flooring computation in $\mathcal{B}_{\mathtt{sk}}$ so that it would cost $3 \times k(k+1)n\mathtt{IMM}$.

- The exact $\mathtt{FastBconvSK}$ to basis q, basically reduces to a fast conversion from $\mathcal{B}$ to $q_{\mathtt{sk}}$, followed by a second one from $b_{\mathtt{sk}}$ to q. So, this is achieved with $3 \times k(k+2)n\mathtt{IMM}$.

- For the relinearization step, since the function $\mathcal{D}_{\mathtt{RNS},q}$ is an automorphism of $\mathcal{R}_q$ data in q can stay in this form throughout the computations. Therefore we can have the vector $\mathcal{D}_{\mathtt{RNS},q}(\boldsymbol{c}_2)$ for free. The two scalar products in $\mathtt{Relin}_{\mathtt{RNS}}$ involve $k^2\mathtt{NTT} + 2k^2n\mathtt{IMM} + 2k\mathtt{invNTT}$, exactly like the relinearisation step in the MP variant.

- Finally $2kn\mathtt{IMM}$ are needed to manage the Montgomery representation, in q, with respect to $\tilde{b}$.

Thus the total cost of the RNS variant is the following:

$$\mathtt{Cost}(\mathtt{Mult}_{\mathtt{RNS}}) = (k^2 + 8k + 4)\mathtt{NTT} + (8k + 3)\mathtt{invNTT} + [10k^2 + 25k + 7]n\mathtt{IMM}.$$

To summarize, the RNS variant decreases the computational cost of the whole homomorphic multiplication algorithm except the parts concerning polynomial multiplications. Also, it involves as many NTT and invNTT as the MP variant. Even by considering an optimized multi-precision multiplication algorithm in the MP variant (with sub-quadratic complexity), the asymptotic computational complexity remains dominated by the $(k^2 + O(k))n\mathtt{NTT}$.

Finally, the MP and RNS variants are asymptotically equivalent when $n \to +\infty$, but the pure RNS variant is more flexible in terms of parallelization.

2.4 Software implementation

The C++ NFLlib library [AMBG+16] was used for efficiently implementing the arithmetic in $\mathcal{R}_q$. It provides an efficient NTT-based product in $\mathcal{R}_q$ for q a product of 30 or 62-bit prime integers, and with degree n a power of 2, up to 2^{15}.

2.4.1 Concrete examples of parameter settings

In this part, we analyze which depth can be reached in a multiplicative tree, and for which parameters. We recall that the initial noise is at most $V = B_{err}(1 + 2\delta_{\mathcal{R}}B_{key})$ and that the output of a tree of depth L has a noise bounded by $C^L_{\mathtt{RNS},1}V + C_{\mathtt{RNS},2}(C^L_{\mathtt{RNS},1} - 1)/(C_{\mathtt{RNS},1} - 1)$ (cf. section 1.2.2) with, for the present RNS variant:

$$\begin{cases} C_{\mathtt{RNS},1} &= \delta_{\mathcal{R}} t \left((1+\rho)(1+\delta_{\mathcal{R}}B_{key}) + 3\right) + \dfrac{\delta_{\mathcal{R}}}{2} \\ C_{\mathtt{RNS},2} &= |q|_t \delta_{\mathcal{R}} t \left((1+\rho)\dfrac{1+\delta_{\mathcal{R}}B_{key}}{2} + \dfrac{5}{2}\right) + (1 + \delta_{\mathcal{R}}B_{key} + \delta^2_{\mathcal{R}}B^2_{key})\left(k + \dfrac{1}{2}\right) \\ & \quad + k\delta_{\mathcal{R}}B_{err}2^{\nu+1}. \end{cases} \tag{2.4.1}$$

We denote by:

$$L_{\mathtt{RNS}} = \max\left\{L \in \mathbb{N} \mid C^L_{\mathtt{RNS},1}V + C_{\mathtt{RNS},2}\frac{C^{L-1}_{\mathtt{RNS},1} - 1}{C_{\mathtt{RNS},1} - 1} \leqslant \frac{q}{t}\left(\frac{1}{2} - \frac{k}{\gamma}\right) - \frac{|q|_t}{2}\right\}$$

the maximal depth allowed by $\texttt{Mult}_{\texttt{RNS}}$, when $\texttt{Dec}_{\texttt{RNS}}$ is used for decryption.

For an 80-bit security level and parameters $B_{key} = 1$, $\sigma_{err} = 8$, $B_{err} = 6\sigma_{err}$, we consider the security analysis in [LN14], which provides ranges for $(\log_2(q), n)$ (cf. [LN14], Tab. 2). We analyze our parameters by using the size of moduli given in `NFLlib`, because those were used for our implementation. For a 32-bit (resp. 64) implementation, a set of 291 30-bit (resp. 1000 62-bit) moduli is available. These moduli are chosen to enable an efficient modular reduction (cf. [AMBG+16], Algorithm 2). Table 3 lists parameters when q and $\mathcal{B}$ are built with these moduli. Note that in powers-of-two cyclotomic, the expansion factor $\delta_{\mathcal{R}}$ is well-known and is equal to the degree of the cyclotomic n. The parameters were determined by choosing the largest ρ (up to $2k-1$), and thus the smallest $\tilde{b}$ (chosen as a power of two for efficiency reasons), allowing to reach the depth $L_{\texttt{RNS}}$ while $L_{\texttt{std}}$ corresponds to the bounds for a classical approach. It also provides sufficient sizes for γ and $b_{\texttt{sk}}$ which allow for $b_{\texttt{sk}}$, to have $|\mathcal{B}| = k$ through (2.2.9) after having chosen for q the k greatest available moduli. One can notice that the bound on γ is smaller than 2^{10} for all cases, thus we can choose γ and t as a power of two with γt fitting in one machine word. For these specific parameters, and despite additional terms in the noise, the multiplicative depth remains unchanged for the `RNS` variants.

n	t	$\tilde{b}$	30-bit moduli					62-bit moduli				
			k	$L_{\texttt{RNS}}$	ρ	$\lceil\log_2(b_{\texttt{sk}})\rceil$	γ	k	$L_{\texttt{RNS}}$	ρ	$\lceil\log_2(b_{\texttt{sk}})\rceil$	γ
2^{11}	2	2^8	3	2 (2)	5	18	7	1	0 (0)	–	–	3
	2^{10}			1 (1)	5	27	7		0 (0)	–	–	3
2^{12}	2	2^8	6	5 (5)	11	21	13	3	4 (4)	5	19	7
	2^{10}			4 (4)	4	27	13		3 (3)	5	28	7
2^{13}	2	2^8	13	13 (13)	1/3	15	46	6	11 (11)	1	17	20
	2^{10}			9 (9)	9	30	27		8 (8)	6	29	33
2^{14}	2	2^8	26	25 (25)	1/2	17	53	12	23 (23)	1/3	16	27
	2^{10}			19 (19)	1	27	53		17 (17)	3	29	34
2^{15}	2	2^{16}	53	50 (50)	1/12	17	351	25	47 (47)	1/2	18	83
	2^{10}			38 (38)	1/2	27	107		37 (37)	1/10	26	255

Table 3: Parameters for `RNS` variants, using the moduli of `NFLlib`. Value in parenthesis correspond to the multiplicative depth for the non-RNS variant.

2.4.2 Influence of $\tilde{b}$ on the noise growth

After a fast conversion from q, ciphertexts in $\mathcal{B}_{\texttt{sk}}$ can contain q-overflow and verify $\|\texttt{ct}'_i\|_\infty < \frac{q}{2}(1+\tau)$ with $\tau \leqslant 2k-1$. By applying Algorithm 15, this bound decreases to $\frac{q}{2}(1+\rho)$, for some $0 < \rho \leqslant 2k-1$. Having $\rho = 2k-1$ in Tab. 3 means that it is unnecessary to use $\texttt{SmMRq}_{\tilde{b}}$ to obtain a best depth, this case only occurs three times. Thus, most of the time doing such reduction is necessary before a multiplication in order to reach the best possible depth. Moreover, choosing a lower ρ (i.e. a higher $\tilde{b}$) than necessary can decrease the lower bound on γ and $b_{\texttt{sk}}$. To illustrate the impact of $\texttt{SmMRq}_{\tilde{b}}$, Fig. 5 depicts the noise growth for $n = 2^{13}$, $t = 2$ and $\tilde{b} \in \{0, 2^8, 2^{16}\}$. According to Tab. 3, $\tilde{b} = 2^8$ is well sufficient in such scenario in order to reach $L_{\texttt{RNS}} = 13$. In comparison, a computation with no reduction at all ($\tilde{b} = 0$) leads to a theoretical depth $L_{\texttt{RNS}} = 11$ in this case. More precisely, choosing $\tilde{b} = 2^8$ implies an average

reduction of 25% of the noise, while with $\tilde{b} = 2^{16}$, we gain around 32%. Consequently, $\texttt{SmMRq}_{\tilde{b}}$ has been systematically integrated within the implementation of $\texttt{Mult}_{\texttt{RNS}}$ in our prototype.

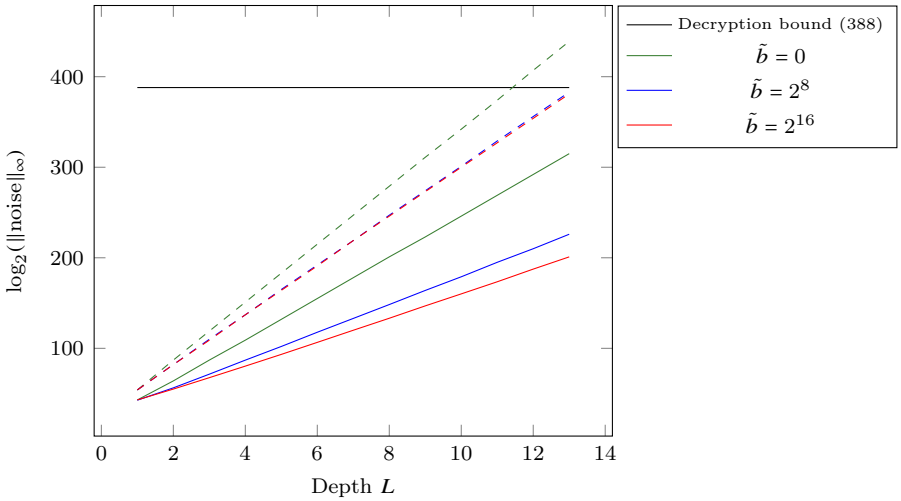


Figure 5: Example of noise growth, for $n = 2^{13}$, $\log_2(q) = 390$ ($\nu = 30, k = 13$), $t = 2$, $\sigma_{err} = 8$, $B_{key} = 1$ (dashed line: theoretical bound, using (2.4.1); plain line: measurements).

2.4.3 Some remarks

Tested algorithms The code[1] we compared with is an open source implementation of `FV` which has been developed in the context of HEAT [2] and which is also based on `NFLlib`. Multi-precision arithmetic is handled with `GMP` 6.1.0 [Gt15], and multiplications by t/q are performed with integer divisions. $\texttt{Mult}_{\texttt{MP}}$ and $\texttt{Dec}_{\texttt{MP}}$ denote functions from this code.

Even if the use of $\texttt{SmMRq}_{\tilde{b}}$ could be avoided in $\texttt{Mult}_{\texttt{RNS}}$ to reach the maximal theoretical depth for some parameters, it has been systematically used since its cost is negligible and it enables a noticeable decrease of noise growth in most of the cases. Moreover relinearization procedure was systematically added at the end of the multiplication, therefore the timings presented for the multiplication correspond in reality to multiplication plus relinearization.

Two variants of $\texttt{Dec}_{\texttt{RNS}}$ (cf. section 2.1.5) have been implemented. Depending on the moduli size ν, the one with floating point arithmetic (named $\texttt{Dec}_{\texttt{RNS-flp}}$ thereafter) uses `double` (resp. `long double`) for double (resp. quadruple) precision, and then does not rely on any other external library at all.

Convenient $\tilde{b}$ and γ Given values of ρ in Tab. 3, $\tilde{b} = 2^8$ (resp. $\tilde{b} = 2^{16}$) satisfies, by far, any set of analyzed parameters. This enables an efficient and straightforward modular arithmetic through standard types like `uint8_t` (resp. `uint16_t`) and a type conversion towards the signed `int8_t` (resp. `int16_t`) immediately gives the centered remainder. The parameter analysis with such $\tilde{b}$ shows that $\gamma = 2^8$ (resp. $\gamma = 2^{10}$) is sufficient to ensure a correct

[1] `https://github.com/CryptoExperts/FV-NFLlib`
[2] https://heat-project.eu/

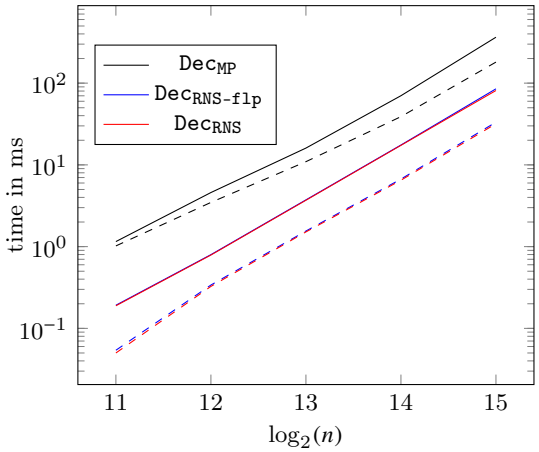


Figure 6: Decryption time ($t = 2^{10}$), with $\nu = 30$ (plain lines) and $\nu = 62$ (dashed lines).

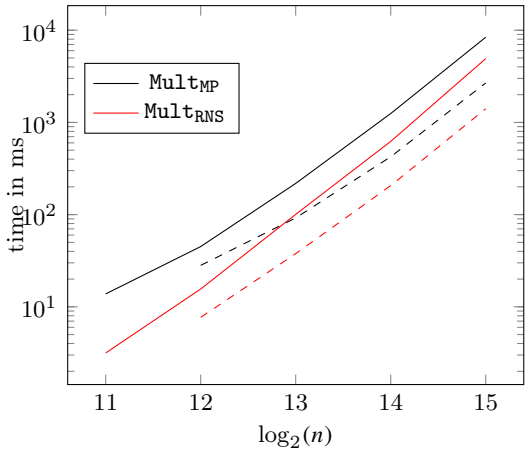


Figure 7: Multiplication time ($t = 2^{10}$), with $\nu = 30$ (plain lines) and $\nu = 62$ (dashed lines).

decryption for all configurations. Furthermore, a reduction modulo γ could be achieved by a simple type conversion to `uint8_t` when $\gamma = 2^8$.

2.4.4 Results

The tests have been run on a laptop, using Fedora 22 with Intel® Core™ i7-4810MQ CPU @ 2.80GHz and using GNU compiler g++ version 5.3.1 with Hyper-Threading and Turbo Boost turned off. The timings corresponds to an average time on 2^{12} decryptions/multiplications.

Figure 6 presents timings for $\texttt{Dec}_{\texttt{MP}}$, $\texttt{Dec}_{\texttt{RNS}}$ and $\texttt{Dec}_{\texttt{RNS-flp}}$, and Fig. 7 depicts timings for $\texttt{Mult}_{\texttt{MP}}$ and $\texttt{Mult}_{\texttt{RNS}}$. Both figures gather data for two modulus sizes: $\nu = 30$ and $\nu = 62$. Relinearization in $\texttt{Mult}_{\texttt{MP}}$ uses a decomposition in radix-base $\omega = 2^{32}$ when $\nu = 30$, and $\omega = 2^{62}$ when $\nu = 62$. The auxiliary bases $\mathcal{B}_{sk}$ and $\mathcal{B}'$ involved in $\texttt{Mult}_{\texttt{RNS}}$ and $\texttt{Mult}_{\texttt{MP}}$ contain $k + 1$ moduli each. Multiplication timing for $(n, \nu, k) = (2^{11}, 62, 1)$ is not given since $L = 1$ already causes decryption failures. The contributions appear in the speed-ups between RNS and MP variants.

In Fig. 6, the two variants of decryption described in 2.1.5 are almost equally fast. Indeed, they perform the same number of elementary (floating point or integer) operations. All the implemented decryption functions take as input a ciphertext in `NTT` representation. Thus, only one `invNTT` is performed (after the product of residues) within each decryption. As explained in section 2.1.5, despite a better asymptotic computational complexity for RNS decryption, the efficiency remains in practice highly related to this `invNTT` procedure, even maybe justifying the slight convergence between MP and RNS decryption times observed. In Fig. 7, the convergence of complexities of $\texttt{Mult}_{\texttt{RNS}}$ and $\texttt{Mult}_{\texttt{MP}}$ (as explained in Sect. 2.3.4) is noticeable.

Table 4 presents detailed timings and speed-ups of RNS vs MP variants together with timings of an RNS decryption and multiplication including the use of SIMD (Single Instruction Multiple Data). Indeed in RNS variants (as well decryption as multiplication), the replacement of division and rounding by base conversions (i.e. matrix-vector multiplications) allows

to benefit, easily and naturally, from concurrent computation. An RNS vector-matrix multiplication naturally owns two levels of parallelization: along the RNS channels, and along the dimension of the result. In `NFLlib`, an element of $\mathcal{R}_q$ is stored in a (32-byte aligned) array `_data` in which the n first values are the coefficients of the polynomial in $\mathcal{R}_{q_1}$, and so forth and so on. Advanced Vector Extensions (`AVX2`) have been used to accelerate the computations.

An `AVX2` register is handled (for our purpose) through the type `__m256i`. This enables to handle either 8x32-bit, or 4x64-bit, or again 16x16-bit integers concurrently. Given the configuration of the `_data` array, reading/writing communications between 256-bit `AVX2` registers and `_data` are the most efficient (through `_mm256_store_si256` and `_mm256_load_si256` intrinsics, which require the 32-byte alignment of `_data`) when the base conversion is parallelized along the dimension n. Therefore this is how we have implemented and tested it. However, since more convenient intrinsic instructions are available for 32-bit, we have made it only within the 32-bit implementations.

Regarding the timings, the impact of `AVX2` remains quite moderate. This is because it is used to accelerate the parts of algorithms besides the `NTT`-based polynomial products which constitute the main cost. For decryption, the RNS variants, floating point and integer arithmetic, are already very efficient, whereas the performance of `AVX2` variant depends on time-consuming loading operations from/to vectorial registers, explaining the small differences. As expected for the multiplication, the timings are converging when n grows, because of the cost of `NTT`'s.

2.5 Conclusion

This work has turned the `FV` scheme entirely compatible with Residue Number Systems and has led to a publication in the 23$^{\text{rd}}$ Selected Areas of Cryptography (SAC) conference ([BEHZ17]). Prior to this work, RNS was already used to accelerate polynomial additions and multiplications. However, the decryption and the homomorphic multiplication involve operations at the coefficient level which were hardly compatible with RNS, such as division and rounding. Our solutions overcome these incompatibilities, without modifying the security features of the original scheme. As a consequence, we have provided a SHE scheme which only involves RNS arithmetic and thus only single-precision integer arithmetic. Moreover the new variant has an higher parallelization potential, since it fully benefits from the properties of RNS, which could be used in future efficient implementations.

The C++ implementation of our prototype has highlighted the gain in practice of our approach compared to the classical one which uses multi-precision arithmetic. Since arithmetic on polynomials is not concerned by this work, our implementation, like the one we are comparing with, has been based on the `NFLlib` library, which embeds a very efficient `NTT`-based polynomial product. Hence, our decryption (resp. homomorphic multiplication) variants offers speed-ups from 20 to 5 (resp. 4 to 2) for cryptographic parameters ensuring 80-bit of security. This work has recently been incorporated by Microsoft Research in the version 2.3 of SEAL, their C++ library implementing FV ([CHH+17]). More recently in [HPS18], Halevi

n	ν	k	variant	Decryption (ms)	Speed-up	Multiplication (ms)	Speed-up
2^{11}	30	3	MP	1.153	–	13.809	–
			RNS-flp	0.192	6.005	–	–
			RNS	0.189	6.101	3.159	4.371
			RNS-AVX2	0.188	6.133	2.710	5.096
	62	1	MP	1.020	–	–	–
			RNS-flp	0.054	18.880	–	–
			RNS	0.050	20.390	–	–
2^{12}	30	6	MP	4.587	–	45.055	–
			RNS-flp	0.798	5.748	–	–
			RNS	0.789	5.814	15.614	2.886
			RNS-AVX2	0.775	5.919	13.737	3.280
	62	3	MP	3.473	–	28.168	–
			RNS-flp	0.339	10.245	–	–
			RNS	0.326	10.653	7.688	3.664
2^{13}	30	13	MP	16.051	–	218.103	–
			RNS-flp	3.732	4.301	–	–
			RNS	3.691	4.349	100.625	2.167
			RNS-AVX2	3.637	4.413	88.589	2.462
	62	6	MP	10.945	–	92.093	–
			RNS-flp	1.552	7.052	–	–
			RNS	1.513	7.234	37.738	2.440
2^{14}	30	26	MP	70.154	–	1,249.400	–
			RNS-flp	17.497	4.009	–	–
			RNS	17.333	4.047	622.596	2.007
			RNS-AVX2	16.818	4.171	617.846	2.022
	62	12	MP	38.910	–	424.014	–
			RNS-flp	6.702	5.806	–	–
			RNS	6.494	5.992	206.511	2.053
2^{15}	30	53	MP	364.379	–	8,396.080	–
			RNS-flp	85.165	4.279	–	–
			RNS	81.225	4.486	4,923.220	1.705
			RNS-AVX2	72.665	5.015	5,063.920	1.658
	62	25	MP	180.848	–	2,680.535	–
			RNS-flp	33.310	5.429	–	–
			RNS	31.895	5.670	1,406.960	1.905

Table 4: Timing results.

et al. proposed another version of this work in which they have chosen to correct the overflows due to base conversions by computing it with double precision floating-point arithmetic resulting in the same noise growth as the original version.

Beyond the immediate benefit for the FV scheme, this work could also be applied to any protocol or scheme relying on primitives using the same kind of operations in a (Ring)-LWE like setting ([BLLN13], [KGV16], [BDF17]).

Chapter 3

Polynomial arithmetic in general cyclotomic rings

Contents

As illustrated in the previous chapters, the bottleneck of Ring-LWE based homomorphic schemes coincides with the one of the underlying arithmetic which is the multiplication of elements. Consequently, because of the efficiency of the NTT with the negative-wrapped convolution technique (Section1.3.2), and also the simplicity of the error sampling in this case (Section1.1.3), most of the research on Ring-LWE based cryptography has focused on power-of-two cyclotomics ($\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$). However despite these advantages, computations in high dimensions (required for FHE for instance), remain hardly practical. As explained in Section 1.3.3, one way to amortize the cost of these operations is to use the batching technique to pack plaintexts. However it is not available for power-of-two cyclotomics with binary plaintext data types. Hence considering other cyclotomic polynomials could offer, beyond a possible greater number of slots for batching, more flexibility regarding to the dimension of the underlying lattice and thus to the security of the Ring-LWE instantiation. Even though such cyclotomics have been previously considered ([GHS12b], [LPR13], [LN14], [SV14], ...), they remain very much unused because, in part, of its less convenient arithmetic.

Yet, in [LPR13], Lyubashevsky et al. have shown how to handle general Ring-LWE instantiations, whether it concerns the associated arithmetic or the error sampling. Their approach relies on the tensored representation of cyclotomic fields ($\mathcal{K}_m = \bigotimes_{i=1}^{r} \mathcal{K}_{m_i}$ where $m = m_1 \cdots m_r$ is the prime power decomposition of m), which allow them to evaluate efficiently (in time $\mathcal{O}(n \log(n))$) the canonical embedding (cf. Section 1.1.2) and its inverse. Once evaluated on the canonical embedding, additions and multiplications of elements are performed coefficient-wise and the error can be sampled from a spherical discrete gaussian, while the inversion of the embedding allows to return in the coefficient representation. Since this method corresponds to an evaluation/interpolation of the elements (seen as polynomials) on the roots of the cyclotomic, the result does not need to be reduced modulo $\mathbf{\Phi}_m$. Hence their method can be seen, somehow, as the generalization of the NTT with the negative-wrapped convolution technique. However, despite its numerous advantages in theory (no reduction modulo $\mathbf{\Phi}_m$, no zeroes-padding to the next power of two), in practice the evaluation of the canonical embedding with this method uses Bluestein and Rader algorithms for FFT ([RSR69]) whose efficiency heavily depends on the prime factors of m and require anyway a zero-padding. Although, an Haskell implementation of their method is available in the $\Lambda \circ \lambda$ library ([CP15]), their method remain seldomly used and several works tend to keep using the univariate representation.

In this context, the work presented in this chapter focuses on improving the efficiency of the arithmetic associated with the "classical" univariate representation in general cyclotomic rings $\mathcal{R} = \mathbb{Z}[X]/(\mathbf{\Phi}_m(X))$ so that one can use only efficient and well-known algorithms like Cooley-Tukey based NTT. Our improvements concern the polynomial reduction required after having performed a product with NTT of size $N = \mathcal{N}_2(2n)$ and not the product itself. We have also highlighted the applicability of our reduction algorithms to two of the most used homomorphic encryption schemes, namely BGV ([BGV12]) and FV ([FV12]) implemented respectively in HELib ([HS14]) and SEAL 2.3 ([CHH+17]).

3.1 Preliminaries

This section aims to familiarize the reader with some elements required when working with general cyclotomic polynomials. We start with the generic Barrett reduction algorithm which is required to compute the reduction modulo $\mathbf{\Phi}_m$ after having performed a polynomial product. We also discuss about the choice of the cyclotomic polynomial in order to maximize: the security, the arithmetic efficiency, and the number of slots.

3.1.1 Barrett's reduction

As explained in Section 1.3.2, once we can compute the polynomial product $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b} \in \mathbb{Z}[X]/(X^N - 1)$ of two elements $(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}_q^2$ through NTT of size N on each RNS channel of $q = q_1 \cdots q_k$ (with q_i prime and congruent to 1 modulo N) through Equation(1.3.8) recalled herafter:

$$\mathrm{NTT}_N(\boldsymbol{c}) = \mathrm{NTT}_N(\boldsymbol{a}) \odot \mathrm{NTT}_N(\boldsymbol{b})$$

Hence, we obtain a polynomial $\boldsymbol{c}$, in `NTT` representation, of degree smaller than $N = \mathcal{N}_2(2n)$, the next power of two greater than, or equal to $2n$. At this point, one needs to reduce $\boldsymbol{c}$ modulo $\mathbf{\Phi}_m$ to get the result back on each $\mathcal{R}_{q_i}$ and therefore in $\mathcal{R}_q$. One way to perform this reduction is to use the generic Barrett algorithm, like in [DS16].

Indeed, Barrett's strategy for modular reduction over the integers [Bar86] can be adapted to polynomial modular arithmetic to reduce a polynomial $\boldsymbol{c}$ of degree $n + \alpha$, with $\alpha \geq 0$, by a polynomial of degree n (for instance $\mathbf{\Phi}_m$) and is presented in Algorithm 17. It consists in computing the quotient of the Euclidean division $\lfloor \boldsymbol{c}/\mathbf{\Phi}_m \rfloor$ through multiplications by precomputed constants and shifts, and once the quotient computed deduce the remainder.

Algorithm 17 NTT based Barrett reduction in $\mathbb{F}_{q_i}[X]$, for a prime $q_i \equiv 1 \bmod N$

Require: $\boldsymbol{c}_{\mathrm{NTT}} = \mathrm{NTT}_N(\boldsymbol{c}) \in \mathbb{F}_{q_i}^N$ with $\deg(\boldsymbol{c}) = n + \alpha < 2n$ with q_i prime, $n = \deg(\mathbf{\Phi}_m)$, $\tilde{n} = \mathcal{N}_2(n)$, $A = \mathcal{N}_2(2\alpha + 1)$, precomputed $NTT_{\tilde{n}}(\mathbf{\Phi}_m)$ and $NTT_A(\lfloor X^{n+\alpha}/\mathbf{\Phi}_m \rfloor)$.
Ensure: $\boldsymbol{c} \bmod (q, \mathbf{\Phi}_m)$ in power-basis.
  function `NTTBarr`$(\boldsymbol{c}_{\mathrm{NTT}}, q_i, \mathbf{\Phi}_m)$
    $\boldsymbol{c} \leftarrow \mathrm{NTT}_N^{-1}(\boldsymbol{c}_{\mathrm{NTT}})$
    $\boldsymbol{f} \leftarrow \lfloor \boldsymbol{c}/X^n \rfloor$
    $\boldsymbol{r} \leftarrow \mathrm{NTT}_A^{-1}\left(\mathrm{NTT}_A(\boldsymbol{f}) \odot \mathrm{NTT}_A(\lfloor X^{n+\alpha}/\mathbf{\Phi}_m \rfloor)\right)$
    $\boldsymbol{r}' \leftarrow \lfloor \boldsymbol{r}/X^{\alpha} \rfloor$
    $\boldsymbol{d} \leftarrow \mathrm{NTT}_{\tilde{n}}^{-1}\left(\mathrm{NTT}_{\tilde{n}}(\boldsymbol{r}') \odot \mathrm{NTT}_{\tilde{n}}(\mathbf{\Phi}_m)\right)$
    $\boldsymbol{c}' \leftarrow \boldsymbol{c} \bmod X^{\tilde{n}} - 1$
    return $\boldsymbol{c}' - \boldsymbol{d}$

Remark 3.1.1. In [DS16], the authors use somehow `NTT`s of size $N = \mathcal{N}_2(2n)$ to perform the second product while in fact it only requires `NTT`s of size $\tilde{n} = \mathcal{N}_2(n)$.

For the sake of completness we provide a proof of correctness of Algorithm 17 just below.

Proof. Since $\mathbb{F}_{q_i}[X]$ is an Euclidean ring, we can write the Euclidean division of $\boldsymbol{c}$ of degree $n+\alpha$ by $\boldsymbol{\Phi}_m$ of degree n:

$$\boldsymbol{c} = \lfloor \boldsymbol{c}/\boldsymbol{\Phi}_m \rfloor \boldsymbol{\Phi}_m + \boldsymbol{r}$$

with $\deg \boldsymbol{r} \leq n-1$. Let $a, b \geq 0$ be two integers, then we can write the following over the field of fractions of $\mathbb{F}_{q_i}[X]$:

$$\frac{X^{n+a}}{\boldsymbol{\Phi}_m} \cdot \frac{\boldsymbol{c}}{X^{n-b}} = \left(\left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m} \right) \cdot X^{a+b}$$

Let $\boldsymbol{r}_1 = |X^{n+a}|_{\boldsymbol{\Phi}_m(X)}$ and $\boldsymbol{r}_2 = |\boldsymbol{c}|_{X^{n-b}}$, we have $\deg(\boldsymbol{r}_1), \deg(\boldsymbol{r}_2) < n$ and we can write:

$$\begin{aligned} &\left(\left\lfloor \frac{X^{n+a}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}_1}{\boldsymbol{\Phi}_m} \right) \cdot \left(\left\lfloor \frac{\boldsymbol{c}}{X^{n-b}} \right\rfloor + \frac{\boldsymbol{r}_2}{X^{n-b}} \right) = \left(\left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m} \right) \cdot X^{a+b} \\ \Leftrightarrow &\left\lfloor \frac{X^{n+a}}{\boldsymbol{\Phi}_m} \right\rfloor \cdot \left\lfloor \frac{\boldsymbol{c}}{X^{n-b}} \right\rfloor + \boldsymbol{r}_a + \boldsymbol{r}_{\alpha+b} + \boldsymbol{r}' = \left(\left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m} \right) \cdot X^{a+b} \end{aligned}$$

with $\deg(\boldsymbol{r}_a) < a$, $\deg(\boldsymbol{r}_{\alpha+b}) < \alpha + b$ and $\deg(\boldsymbol{r}') < 0$. Writing the Euclidean division of the product on the left part by X^{a+b} leads to:

$$\left\lfloor \frac{\lfloor X^{n+a}/\boldsymbol{\Phi}_m \rfloor \cdot \left\lfloor \boldsymbol{c}/X^{n-b} \right\rfloor}{X^{a+b}} \right\rfloor X^{a+b} + \boldsymbol{r}'' + \boldsymbol{r}_a + \boldsymbol{r}_{\alpha+b} + \boldsymbol{r}' = \left(\left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m} \right) \cdot X^{a+b}$$

with $\deg(\boldsymbol{r}'') < a + b$, and thus we finally obtain:

$$\Leftrightarrow \left\lfloor \frac{\lfloor X^{n+a}/\boldsymbol{\Phi}_m \rfloor \cdot \left\lfloor \boldsymbol{c}/X^{n-b} \right\rfloor}{X^{a+b}} \right\rfloor + \frac{\boldsymbol{r}'' + \boldsymbol{r}_a + \boldsymbol{r}_{\alpha+b} + \boldsymbol{r}'}{X^{a+b}} = \left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m}$$

By choosing $b \geqslant 0$ and $a \geqslant \alpha$, the right term of the left member of the equation above have a degree smaller than 0. Therefore we can deduce that the two floored polynomials are equal.

Hence, by taking $b = 0$ and $a = \alpha$, we get that $\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \rfloor$ is equal to the flooring of the left part of last equation, which is what Algorithm 17 computes for $\boldsymbol{r}'$. Indeed, since $\lfloor X^{n+\alpha}/\boldsymbol{\Phi}_m \rfloor \cdot \lfloor \boldsymbol{c}/X^n \rfloor$ is of degree strictly smaller than $2\alpha + 1$, the computation can be done with an NTT of size $A = \mathcal{N}_2(2\alpha + 1)$.

Finally, we notice that the result of the computation of $\boldsymbol{r} = \boldsymbol{c} - \lfloor \boldsymbol{c}/\boldsymbol{\Phi}_m \rfloor \times \boldsymbol{\Phi}_m$ has a degree strictly smaller than n. Since the polynomial $\boldsymbol{c}'$ in the algorithm is nothing but $\lfloor \boldsymbol{c}/\boldsymbol{\Phi}_m \rfloor \times \boldsymbol{\Phi}_m \bmod X^{\tilde{n}} - 1$ (the reduction modulo $X^{\tilde{n}} - 1$ is a consequence of the NTT based polynomial product in dimension $\tilde{n}$) at the end we have that $\boldsymbol{c}' - \boldsymbol{d} = (\boldsymbol{c} - \lfloor \boldsymbol{c}/\boldsymbol{\Phi}_m \rfloor \times \boldsymbol{\Phi}_m) \bmod (X^{\tilde{n}} - 1) = \boldsymbol{c} \bmod \boldsymbol{\Phi}_m$. The last equality holds because $\deg(\boldsymbol{c} \bmod \boldsymbol{\Phi}_m) < n$ and is thus strictly smaller than $\tilde{n}$. □

One can notice, that the performance of Algorithm 17 is directly related to the size of the polynomial to be reduced: the algorithm is more efficient when α is small. More precisely if

we denote $\tilde{n} = \mathcal{N}_2(n)$ and $A = \mathcal{N}_2(2\alpha + 1)$ the cost of the algorithm is given by:

$$\mathcal{C}_{\mathrm{NTT}}(N) + 2\mathcal{C}_{\mathrm{NTT}}(\tilde{n}) + 2\mathcal{C}_{\mathrm{NTT}}(A) + (\tilde{n} + A)\mathtt{MMult}_{q_i}$$

where $\mathcal{C}_{\mathrm{NTT}}(x)$ denotes the cost for evaluating (1.3.7) (or its inverse) of size x and $\mathtt{MMult}_{q_i}$ the cost of a modular multiplication modulo q_i.

3.1.2 About the choice of the cyclotomic

As shown in Section 1.3.2, the bottleneck of Ring-LWE arithmetic is the polynomial products whose asymptotic complexity is quasi-linear in the degree n of $\mathbf{\Phi}_m$, and even more so when we are not using power-of-two cyclotomics and thus NTT with negative-wrapped convolution. In practice, the most efficient NTT algorithms require to have a degree which is a power-of-two. Hence, even if it requires to pad the polynomials' coefficients with zeros from $2n$ to N, we use NTTs of size $N = \mathcal{N}_2(2n)$ the next power of two greater than or equal to $2n$. Since practical security of Ring-LWE instantiations increases with the degree n (for a modulus q and an error distribution χ_{err} carefully chosen, see Section 4.2.5), choosing n (and thus $2n$) directly as a power-of-two maximizes the security without affecting the efficiency of the underlying arithmetic.

Lemma 3.1.2 ([DF04], Preliminaries)**.** Let $m = p_1^{r_1} \cdots p_s^{r_s}$ be the prime decomposition of m where each $r_i \geq 1$, then

$$\varphi(m) = \prod_{i=1}^{s} (p_i - 1) p_i^{r_i - 1} \tag{3.1.1}$$

where φ denotes the Euler-totient function.

According to Equation(3.1.1), in order to have $n = \varphi(m)$ equal to a power of two we must select m such that $m = 2^k p_1 \cdots p_s$, where the factors p_i are prime numbers of the shape $p_i = 2^{l_i} + 1$. It is proven that l_i must necessary be a power-of-two for such number to be prime, in this case these numbers are called Fermat numbers $(2^{2^{n_i}} + 1)$. However this condition is not sufficient for the primality and it is not known whether or not there are an infinite number of Fermat primes. Actually, so far we only know 5 such numbers, which correspond to the first five Fermat numbers: $F_0 = 3$, $F_1 = 5$, $F_2 = 17$, $F_3 = 257$ and $F_4 = 65537$.

Lemma 3.1.3 ([Lan05], chapter 6.3)**.** Let $m = 2^k r$ with r an odd integer, then

$$\mathbf{\Phi}_m(X) = \mathbf{\Phi}_r(-X^{2^{k-1}}) \tag{3.1.2}$$

Equation (3.1.2) shows that a cyclotomic of index $m = 2^k r$ with r odd, offers as many batching slots (see Section 1.3.3) than the cyclotomic of index r. Therefore in order to minimize the ratio n/ℓ, with ℓ is the number of batching slots, and thus the arithmetic efficiency one should choose m odd. As a consequence, in order to maximize the degree, and thus the security, without increasing the arithmetic cost, m should be chosen as a power of Fermat primes. Moreover choosing such m allows us to know precisely the number of slots

of $\mathbf{\Phi}_m$ when the plaintext modulus is $t = 2$ which is particularly interesting for evaluating homomorphically boolean circuits for instance.

Lemma 3.1.4. Let $m = p_1 \cdots p_s$ the product of s distinct Fermat primes sorted accordingly to their size, i.e. $p_i = 2^{l_i} + 1$ prime with $l_i = 2^{k_i}$ and $p_1 < p_2 < \ldots < p_s$, then $\mathbf{\Phi}_m$ splits in $\ell = 2^{l_1+\cdots+l_s-k_s-1}$ irreducible factors modulo 2.

Proof. Lemma 1.3.3 states that the number of irreducible factors of $\mathbf{\Phi}_m$ modulo 2 is given by $\ell = n/d$ where $n = \varphi(m)$ and d is the order of 2 modulo m. It is not hard to see that the order of 2 modulo each p_i is $o_i = 2l_i$, and since $\mathbb{Z}/m\mathbb{Z}$ splits in the cartesian product of the $\mathbb{F}_{p_i}$ through the CRT we know that 2 has order $\text{lcm}(o_1, \ldots, o_s)$ modulo m. Moreover $o_1 \mid o_2 \mid \cdots \mid o_s$ therefore 2 has order o_s modulo m and so $\ell = n/o_s = 2^{l_1+\cdots+l_s}/2l_s = 2^{l_1+\cdots+l_s-k_s-1}$. □

Last but not least, it would be interesting to know if cyclotomics whose index is a power of Fermat primes offer good ratios between the practical cost of their arithmetic and their number of slots. Indeed if we quantify the arithmetic cost in practice $N \log_2(N)$ ($\simeq$ complexity of NTTs and reductions algorithms), cyclotomics offering the smallest ratio $N \log_2(N)/\ell$ are those which "amortize" best the cost of the arithmetic.

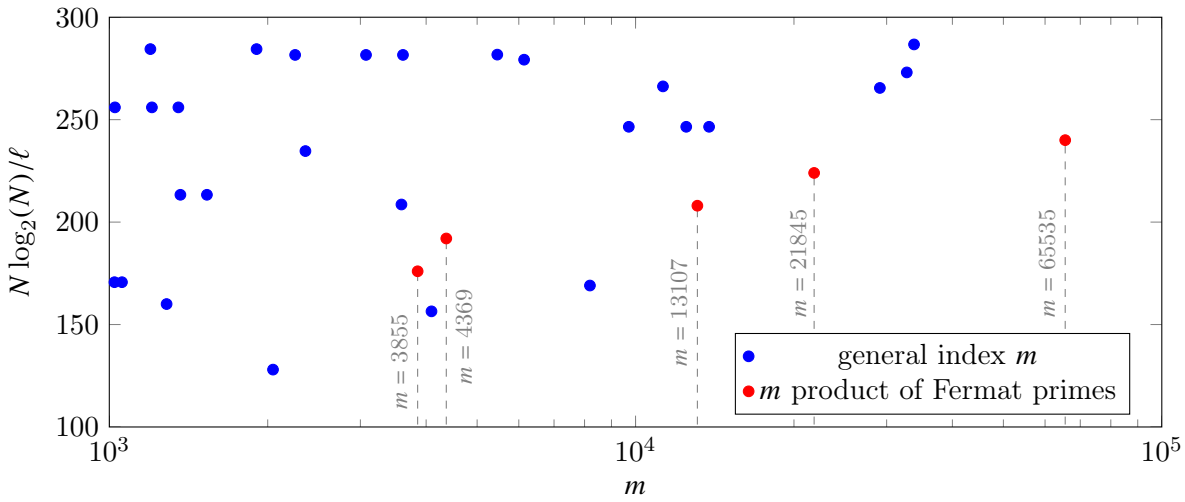


Figure 8: Amortized cost of the arithmetic for cyclotomics of index $1000 \leq m \leq 100000$

Figure 8 indexes $10^3 \leq m \leq 10^5$ of cyclotomic polynomials with the smallest ratios $N \log_2(N)/\ell$. One can see that besides offering a power-of-two degree, and therefore a probably higher level of security for an arithmetic of size $N = \mathcal{N}_2(2n)$, Fermat primes product ratios are among the best ones. Of course, the smallest ratios correspond to the small degree but depending on the targeted application we may need a large degree. For instance with homomorphic encryption, the multiplicative depth increases with the ciphertext modulus q whose size is bounded by the degree n, for a fixed σ_{err} and a given level of security (cf. Equation(1.1.3)). As a consequence large degree cyclotomics are required for evaluating arithmetic circuits with an important multiplicative depth.

3.2 Improving polynomial reduction modulo $\mathbf{\Phi}_m$

In this section, we propose two efficient methods to compute polynomial reductions. The first method takes advantage of the properties of the cyclotomic polynomials to improve the efficiency of the Barrett algorithm. The second reduction rests on an adaptation of the Montgomery modular reduction over integers [Mon85] to polynomials. In order to ease our exposition, throughout the rest of this section, unless stated otherwise, the letter q will refer to a prime congruent to 1 modulo N. The reader should not forget that for applications to Ring-LWE the modulus q is chosen composite $q = q_1 \cdots q_k$ (c.f. Section 1.3.1), and thus products and polynomial reductions need to be done in each RNS channel of q, i.e. modulo each prime factor q_i of q.

3.2.1 Barrett's reduction for cyclotomic polynomials

As explained in the previous section, Barrett algorithm is sensitive to the difference between the degree of the polynomial to be reduced and that of the polynomial we want to reduce by. The smaller the difference, the more efficient the algorithm will be. Herein we propose an efficient method to reduce this difference before doing the actual Barrett reduction.

In our context, polynomials have degree $2n-2$ and coefficients in $\mathbb{Z}/q\mathbb{Z}$ for a prime q congruent to 1 modulo N and have to be reduced modulo $\mathbf{\Phi}_m$. Let $\boldsymbol{c}$ be such a polynomial, if $\boldsymbol{c}$ was reduced by a polynomial $\boldsymbol{Q}_{\text{sp}}$ of degree $n+\alpha+1$, the difference between the degree of the polynomial and the degree of $\mathbf{\Phi}_m$ would drop to α. However, in order to obtain the correct value modulo $\mathbf{\Phi}_m$ in the end, $\mathbf{\Phi}_m$ has to divide $\boldsymbol{Q}_{\text{sp}}$ and for the efficiency of the reduction $\boldsymbol{Q}_{\text{sp}}$ should be sparse enough so that its reduction can be handled through few operations in each $\mathbb{F}_{q_i}$. X^m-1 would be a good candidate however the difference between m and $n=\varphi(m)$ is quite important most of the time, and in particular for the Fermat primes product cyclotomics we consider. Nonetheless, one can find a suitable sparse polynomial $\boldsymbol{Q}_{\text{sp}}$ of degree smaller than m by using cyclotomics properties.

First, the property given in Proposition 1.1.5: $\prod_{d/m} \mathbf{\Phi}_d(X) = X^m - 1$, allows us to choose $\boldsymbol{Q}_{\text{sp}}$ as the product of $\mathbf{\Phi}_m$ and some $\mathbf{\Phi}_d$ for d dividing m.

Lemma 3.2.1 ([Lan05], chapter 6.3)**.** The following propositions are true:

- let $m = p_1^{r_1} \cdots p_s^{r_s}$ be an integer with its prime factorization and let $r = p_1 \cdots p_s$ be the radical of m then: $\mathbf{\Phi}_m(X) = \mathbf{\Phi}_r(X^{m/r})$;
- let p be a prime not dividing m' then: $\mathbf{\Phi}_{m'p}(X) \cdot \mathbf{\Phi}_{m'}(X) = \mathbf{\Phi}_{m'}(X^p)$.

The first statement generalizes Lemma 3.1.3 in the sense that a cyclotomic of index m will have a higher degree, but the same number of slots, than the cyclotomic of index r the radical of m. Hence to optimize the arithmetic efficiency one should choose a square free index m. By using recursively the second statement on a square free index $m = dh$, for some d dividing m, one can choose $\boldsymbol{Q}_{\text{sp}} = \mathbf{\Phi}_d(X^h)$ since we have $\mathbf{\Phi}_m \mid \boldsymbol{Q}_{\text{sp}} \mid X^m - 1$. Therefore one should choose d dividing m such that $\mathbf{\Phi}_d$ is very sparse and its degree $h\varphi(d)$ is as close as possible

to n. Moreover if d has at most two distinct odd prime factors then $\boldsymbol{\Phi}_d$ and thus $\boldsymbol{Q}_{\text{sp}}$ has its coefficients lying in $\{-1, 0, 1\}$ ([Isa94]). In this case reduction modulo $\boldsymbol{Q}_{\text{sp}}$ in $\mathbb{F}_q$ only requires additions modulo q and can be performed very efficiently.

In addition, when $m < 2n-2$, $\boldsymbol{c}$ can initially be reduced by $X^m - 1$ with $2n-m-1$ additions in $\mathbb{F}_q$. Since $\boldsymbol{\Phi}_m(X) \mid \boldsymbol{Q}_{\text{sp}}(X) \mid X^m - 1$ the strategy remains correct, and the complexity of the reduction by $\boldsymbol{Q}_{\text{sp}}(X)$ is further reduced. More precisely, if $\texttt{HW}(\boldsymbol{Q}_{\text{sp}})$ denotes the Hamming weight of $\boldsymbol{Q}_{\text{sp}}$, i.e. its number of non-zero coefficients, the cost of the reduction of $\boldsymbol{c}$ by $\boldsymbol{Q}_{\text{sp}}$ is:

$$(\texttt{HW}(\boldsymbol{Q}_{\text{sp}}) - 1)(m - \deg(\boldsymbol{Q}_{\text{sp}}))$$

additions in each $\mathbb{F}_{q_i}$. At this point, we obtain $\boldsymbol{c'} = \boldsymbol{c} \bmod \boldsymbol{Q}_{\text{sp}}$ (with $\deg(\boldsymbol{c'}) \leq n + \alpha$ and $\boldsymbol{c'} \equiv \boldsymbol{c} \bmod \boldsymbol{\Phi}_m$).

Our optimized Barrett algorithm is depicted in Algorithm 18. It starts by recovering $\boldsymbol{c}$ in power-basis from the NTT representation output by (1.3.8). Then it consecutively reduces $\boldsymbol{c}$ of degree $2n-2$ by $X^m - 1$ and by the sparse polynomial $\boldsymbol{Q}_{\text{sp}}$. This allows to recover $\boldsymbol{c'} = \boldsymbol{c} \bmod \boldsymbol{Q}_{\text{sp}}$ of degree $n + \alpha$ very efficiently. Afterwards, we apply the classical Barrett reduction to $\boldsymbol{c'}$ (given in coefficients representation) to get $\boldsymbol{c''} = \boldsymbol{c} \bmod \boldsymbol{\Phi}_m$.

Algorithm 18 Optimized NTT-based Barrett reduction in $\mathbb{F}_q[X]$, for a prime $q \equiv 1 \bmod N$.

Require: $\boldsymbol{c}_{\text{NTT}} = \text{NTT}_N(\boldsymbol{c})$ with $\deg(\boldsymbol{c}) \leqslant 2n-2$, $\boldsymbol{Q}_{\text{sp}} \in \mathbb{Z}[X]$ of degree $n+\alpha+1$, $N = \mathcal{N}_2(2n)$, $\tilde{n} = \mathcal{N}_2(n)$, $A = \mathcal{N}_2(2\alpha+1)$, precomputed $NTT_{\tilde{n}}(\boldsymbol{\Phi}_m)$ and $NTT_A(\lfloor X^{n+\alpha}/\boldsymbol{\Phi}_m \rfloor)$.
Ensure: $\boldsymbol{c''} = \boldsymbol{c} \bmod \boldsymbol{\Phi}_m$ in power-basis.
  function $\texttt{REDBT}_{\boldsymbol{\Phi}_m}(\boldsymbol{c}_{\text{NTT}}, q, \boldsymbol{\Phi}_m, \boldsymbol{Q}_{\text{sp}})$
    $\boldsymbol{c} \leftarrow \text{NTT}_N^{-1}(\boldsymbol{c}_{\text{NTT}})$
    if $m < 2n-2$ **then**
      $\boldsymbol{c} \leftarrow \boldsymbol{c} \bmod X^m - 1$
    $\boldsymbol{c'} \leftarrow \boldsymbol{c} \bmod \boldsymbol{Q}_{\text{sp}}$
    $\boldsymbol{f} \leftarrow \lfloor \boldsymbol{c}/X^n \rfloor$
    $\boldsymbol{r} \leftarrow NTT_A^{-1}(NTT_A(\boldsymbol{f}) \odot NTT_A(\lfloor X^{n+\alpha}/\boldsymbol{\Phi}_m \rfloor))$
    $\boldsymbol{r'} \leftarrow \lfloor \boldsymbol{r}/X^{\alpha} \rfloor$
    $\boldsymbol{d} \leftarrow NTT_{\tilde{n}}^{-1}(NTT_{\tilde{n}}(\boldsymbol{r'}) \odot NTT_{\tilde{n}}(\boldsymbol{\Phi}_m))$
    $\boldsymbol{c''} \leftarrow \boldsymbol{c} \bmod X^{\tilde{n}} - 1$
    return $\boldsymbol{c''} - \boldsymbol{d}$

The impact of this sparse reduction is illustrated in Table 5, where polynomials $\boldsymbol{Q}_{\text{sp}}$ are presented for different cyclotomic polynomials. We have chosen to use cyclotomics whose index is a product of Fermat primes for the reasons discussed in Section 3.1.2. The number of batching slots ℓ associated with each cyclotomic is also presented. Moreover, since our method can be applied to any index m we have also taken the index $m = 32767 = 7 \times 31 \times 151$ which is not a product of Fermat primes but which has a relatively good ratio $N \log_2(N)/\ell$ (Section 3.1.2). In order to highlight the sparsity of $\boldsymbol{Q}_{\text{sp}}$ we also give $\texttt{HW}(\boldsymbol{Q}_{\text{sp}})$ which is the number of non-zero coefficients of $\boldsymbol{Q}_{\text{sp}}$.

Complexity

Since the complexity of computing multiplications in $\mathbb{F}_q$ is much higher than additions, the cost of the reduction by the sparse polynomial can be neglected. Moreover, with the RNS, each multiplication in $\mathcal{R}_q$, with $q = q_1 \dots q_k$ can be decomposed into k independent and smaller multiplications. The degree of $\boldsymbol{Q}_{\text{sp}}$ is $n + \alpha + 1$ thus NTT of size $A = \mathcal{N}_2(2\alpha + 1)$ are required to compute the first polynomial product in Algorithm 18. This is in contrast with $N = \mathcal{N}_2(2n)$ which would have been the size required without using the sparse reduction. Therefore the cost, in terms of modular multiplications, to reduce the polynomial $\boldsymbol{c}$ output by (1.3.8) is essentially k times the cost of Algorithm 18:

$$k \cdot (C_{\text{NTT}}(N) + 2 \cdot C_{\text{NTT}}(A) + 2 \cdot C_{\text{NTT}}(\tilde{n}) + A + \tilde{n}) .$$

While the cost of the method by using directly Barrett's algorithm, i.e. without performing the reduction by the sparse polynomial, is:

$$k \cdot (3 \cdot C_{\text{NTT}}(N) + 2 \cdot C_{\text{NTT}}(\tilde{n}) + N + \tilde{n}) .$$

Based on this analysis, and assuming that $C_{\text{NTT}}(N) = N \log_2(N)$, we also provide in Table 5 the theoretical speed-up obtained with the use of the sparse reduction. Those theoretical speed-ups can be compared with those obtained in practice which are presented in Table 10.

m	n	ℓ	$\boldsymbol{Q}_{\text{sp}}$	$\deg(\boldsymbol{Q}_{\text{sp}})$	α	$\text{HW}(\boldsymbol{Q}_{\text{sp}})$	A	N	Speed-up
3855	2048	128	$\mathbf{\Phi}_{3\cdot5}(X^{257})$	2056	7	7	2^4	2^{12}	2.06
4369	4096	256	$\mathbf{\Phi}_{17}(X^{257})$	4112	15	17	2^5	2^{13}	2.05
13107	8192	512	$\mathbf{\Phi}_{3}(X^{17\cdot257})$	8738	545	3	2^{11}	2^{14}	1.86
21845	16384	1024	$\mathbf{\Phi}_{5}(X^{17\cdot257})$	17476	1091	5	2^{12}	2^{15}	1.86
32767	27000	1800	$\mathbf{\Phi}_{7}(X^{31\cdot151})$	28086	1085	7	2^{12}	2^{16}	1.95
65535	32768	2048	$\mathbf{\Phi}_{3\cdot5}(X^{17\cdot257})$	34952	2183	7	2^{13}	2^{16}	1.85

Table 5: Sparse polynomials used for partial reduction with their related parameters.

3.2.2 NTT-based Montgomery's reduction

We propose a Montgomery reduction of a polynomial given in NTT representation, inspired by the orginal Montgomery reduction over the integers [Mon85] and the work of Bajard et al. [BIN06]. The purpose of this new reduction is to overcome the bottleneck of the previous optimized Barrett algorithm, which is the computation of the inverse NTT of size N of the input polynomial (1.3.8). Our Montgomery reduction takes advantage of the presence of the NTT basis of size $N/2$ (seen as an RNS basis in [BIN06]) in the basis of size N allowing to perform all the computations, in particular the inverse NTT evaluation, in the basis of size $N/2$ instead of N.

The NTT representation of a polynomial of size N was defined in (1.3.7) as the set $\{\boldsymbol{c} \bmod (X - \psi^j) | 0 \leq j < N\}$ with ψ a primitive N-th root of unity in $\mathbb{F}_q$. This representation can be

seen as a polynomial-RNS representation of $\boldsymbol{c}$ mod $(X^N - 1)$ since $X^N - 1 = \prod_{0 \leq j < N}(X - \psi^j)$ modulo q, with respect to the following NTT-basis:

$$\mathcal{B}_{\psi,N} = \{|X - 1|_q, |X - \psi|_q, \ldots, |X - \psi^{N-1}|_q\}$$

As $X^N - 1$ splits in $(X^{N/2} - 1)(X^{N/2} + 1)$ when N is even, half of the NTT_N representation of $\boldsymbol{c}$ corresponds to its $\text{NTT}_{N/2}$ representation. Hence, the basis $\mathcal{B}_{\psi,N}$ is split along even and odd powers of ψ. We can then define two sub-bases defining two polynomials:

$$\begin{cases} \mathcal{B}^{(e)}_{\psi,N} = \{|X - \psi^{2j}|_q, 0 \leq j \leq \frac{N}{2} - 1\} & \text{and} \quad \boldsymbol{\Psi}^{(e)} = \left|\prod_{j=0}^{N/2-1}(X - \psi^{2j})\right|_q \\ \mathcal{B}^{(o)}_{\psi,N} = \{|X - \psi^{2j+1}|_q, 0 \leq j \leq \frac{N}{2} - 1\} & \text{and} \quad \boldsymbol{\Psi}^{(o)} = \left|\prod_{j=0}^{N/2-1}(X - \psi^{2j+1})\right|_q \end{cases} \tag{3.2.1}$$

It is straightforward to notice that $\boldsymbol{\Psi}^{(e)} \equiv |X^{N/2} - 1|_q$ and $\boldsymbol{\Psi}^{(o)} \equiv |X^{N/2} + 1|_q$. We also note that since N is a power of two, one has $\boldsymbol{\Psi}^{(o)} \equiv \boldsymbol{\Phi}_N \bmod q$. Therefore, thanks to Lemma 3.2.3, whose first point is a direct consequence of Lemma 3.2.2 which is recalled below, we can choose $X^{N/2} + 1$ as the Montgomery factor.

Lemma 3.2.2 ([Fil00], Lemma 2)**.** Let n and m be positive integers with $n > m$. If the quotient n/m is not a power of a prime, then for every integer a, there exist $\boldsymbol{U}$ and $\boldsymbol{V}$ in $\mathbb{Z}[X]$ satisfying:

$$\boldsymbol{\Phi}_m(X) \cdot \boldsymbol{U}(X) + \boldsymbol{\Phi}_n(X) \cdot \boldsymbol{V}(X) = a$$

If for some prime p and some positive integer t we have $n/m = pt$, then there exist $\boldsymbol{U}$ and $\boldsymbol{V}$ in $\mathbb{Z}[X]$ satisfying the above equation if and only if $p \mid a$

Lemma 3.2.3. Let $\boldsymbol{\Phi}_m$ be the m-th cyclotomic polynomial of degree n and N be the smallest power of two greater than or equal to $2n$. If m is not a power of two then:

- there exists $(\boldsymbol{U}, \boldsymbol{V}) \in \mathbb{Z}[X]^2$ such that $\boldsymbol{U}(X) \cdot \boldsymbol{\Phi}_m(X) + \boldsymbol{V}(X) \cdot \boldsymbol{\Phi}_N(X) = 1$;
- for any prime q, $\boldsymbol{\Phi}_m$ and $(X^N - 1)$ are coprime in $\mathbb{F}_q$. In particular $\boldsymbol{\Phi}_m$ is a unit in $\mathbb{F}_q[X]/(\boldsymbol{\Phi}_N)$.

Proof. The first point is a direct consequence from Lemma 3.2.2. Since m is not a power of two, m cannot divide N. By denoting $m = 2^r m'$ with $m' > 1$ an odd integer we have $n = 2^{r-1}\varphi(m')$, thus $2n = 2^r\varphi(m')$ and then if N divides m, $\mathcal{N}_2(\varphi(m')) = 1$ which is not possible since $m' \geqslant 3$. Therefore N and m do not divide each other and we can apply Lemma 3.2.2.

Let α be a root of $\boldsymbol{\Phi}_m$ in the algebraic closure of $\mathbb{F}_q$. If α is also a root of $X^N - 1$ then $\alpha^N = 1$, since α is of order m by definition of $\boldsymbol{\Phi}_m$ it implies that m divides N which is impossible since N is a power of two and m is not. So, $\boldsymbol{\Phi}_m$ and $X^N - 1$ are coprime on the algebraic closure of $\mathbb{F}_q$ thus in $\mathbb{F}_q$. The second point comes from Bezout equality in $\mathbb{F}_q$ and from the fact that $X^N - 1 \equiv (X^{N/2} - 1)(X^{N/2} + 1) \bmod q$. □

One can extract from the coordinates of $\boldsymbol{c}$ in $\mathcal{B}_{\omega,N}$ the representation $\widehat{\boldsymbol{c}}^{(e)}$ of $\boldsymbol{c}$ in $\mathcal{B}^{(e)}_{\omega,N}$ (resp. $\widehat{\boldsymbol{c}}^{(o)}$ in $\mathcal{B}^{(o)}_{\omega,N}$). So, given $\widehat{\boldsymbol{c}}^{(o)}$ and $\widehat{\boldsymbol{c}}^{(e)}$, we can use the NTT operator to get:

$$\mathtt{NTT}^{-1}_{N/2}(\widehat{\boldsymbol{c}}^{(e)}) = \boldsymbol{c} \bmod (q, X^{N/2}-1).$$

Definition 3.2.4. We define the following function, which takes in as input the residues of the polynomial $\boldsymbol{c}$ ($\deg(\boldsymbol{c}) < N$) modulo a prime q:

$$\mathtt{modMg}_{\boldsymbol{\Phi}_m,\boldsymbol{\Psi}^{(o)},q}(\boldsymbol{c}) = \frac{\boldsymbol{c} + \boldsymbol{\Phi}_m \times |-\boldsymbol{c}/\boldsymbol{\Phi}_m|_{\boldsymbol{\Psi}^{(o)}}}{\boldsymbol{\Psi}^{(o)}} \bmod q. \tag{3.2.2}$$

The $\mathtt{modMg}_{\boldsymbol{\Phi}_m,\boldsymbol{\Psi}^{(o)},p}$ function defined in (3.2.2) is a classical Montgomery reduction with factor $\boldsymbol{\Psi}^{(o)}$ and consisting in an exact polynomial division. It always outputs a polynomial congruent to $|\boldsymbol{c}/\boldsymbol{\Psi}^{(o)}|_{\boldsymbol{\Phi}_m}$, furthermore when $\deg(\boldsymbol{c}) \leqslant N/2 + n - 1$ the output is exactly $|\boldsymbol{c}/\boldsymbol{\Psi}^{(o)}|_{\boldsymbol{\Phi}_m}$.

Lemma 3.2.5. If $\deg(\boldsymbol{c}) \leqslant \frac{N}{2} + n - 1$, then $\mathtt{modMg}_{\boldsymbol{\Phi}_m,\boldsymbol{\Psi}^{(o)},p}(\boldsymbol{c}) = \boldsymbol{c}/\boldsymbol{\Psi}^{(o)} \bmod (p, \boldsymbol{\Phi}_m)$.

Proof. The degree of the numerator in (3.2.2) is bounded by $\max(\deg(\boldsymbol{c}), \deg(\boldsymbol{\Phi}_m)+\deg(\boldsymbol{\Psi}^{(o)})-1) \leqslant N/2 + n - 1$. Thus, the degree of the resulting quotient is bounded by $n - 1 < N/2$. Therefore, the output is $|\boldsymbol{c}/\boldsymbol{\Psi}^{(o)}|_{\boldsymbol{\Phi}_m}$ and the computation of (3.2.2) can be made modulo $X^{N/2} - 1$, i.e. in an NTT representation of size $N/2$ when using primes $q \equiv 1 \bmod N$. □

Algorithm 19 details the computation of (3.2.2). The following precomputations are used therein:

$$\begin{cases} \widehat{\boldsymbol{W}}^{(o)} & : & -1/\boldsymbol{\Phi}_m \bmod (q, \boldsymbol{\Psi}^{(o)}) \text{ in base } \mathcal{B}^{(o)}_{\omega,N} \\ \widehat{\boldsymbol{Y}}^{(e)} & : & 1/\boldsymbol{\Psi}^{(o)} \equiv 1/2 \bmod (q, \boldsymbol{\Psi}^{(e)}) \text{ in base } \mathcal{B}^{(e)}_{\omega,N} \\ \widehat{\boldsymbol{Z}}^{(e)} & : & \boldsymbol{\Phi}_m/\boldsymbol{\Psi}^{(o)} \equiv \boldsymbol{\Phi}_m/2 \bmod (q, \boldsymbol{\Psi}^{(e)}) \text{ in base } \mathcal{B}^{(e)}_{\omega,N} \end{cases}$$

Algorithm 19 NTT-based Montgomery reduction in $\mathbb{F}_q[X]$, for a prime $q \equiv 1 \bmod N$

Require: $\widehat{\boldsymbol{c}} = \mathtt{NTT}_{q,N}(\boldsymbol{c})$, with $N = \mathcal{N}_2(2n)$ and $\deg(\boldsymbol{c}) \leqslant 2n-2 < N/2+n-1$ and precomputed constants $\widehat{\boldsymbol{W}}^{(o)}$, $\widehat{\boldsymbol{Y}}^{(e)}$ and $\widehat{\boldsymbol{Z}}^{(e)}$.
Ensure: $\boldsymbol{R} = (\boldsymbol{c}/\boldsymbol{\Psi}^{(o)}) \bmod (q, \boldsymbol{\Phi}_m)$ in power-basis.
1 **function** $\mathtt{REDMG}_{\boldsymbol{\Phi}_m,\boldsymbol{\Psi}^{(o)},q}(\widehat{\boldsymbol{c}}, \widehat{\boldsymbol{Y}}^{(e)}, \widehat{\boldsymbol{Z}}^{(e)})$
2 $(\widehat{\boldsymbol{c}}^{(e)}, \widehat{\boldsymbol{c}}^{(o)}) \leftarrow \mathtt{Split}(\widehat{\boldsymbol{c}})$ ▷ Split the NTT coeff. wrt parity of indexes
3 $\widehat{\boldsymbol{Q}}^{(o)} \leftarrow \widehat{\boldsymbol{c}}^{(o)} \odot \widehat{\boldsymbol{W}}^{(o)}$
4 $\widehat{\boldsymbol{Q}}^{(e)} \leftarrow \mathtt{BaseConv}(\widehat{\boldsymbol{Q}}^{(o)})$ ▷ base conversion from $\mathcal{B}^{(o)}_{\omega,N}$ to $\mathcal{B}^{(e)}_{\omega,N}$
5 $\widehat{\boldsymbol{T}}^{(e)} \leftarrow \widehat{\boldsymbol{c}}^{(e)} \odot \widehat{\boldsymbol{Y}}^{(e)} + \widehat{\boldsymbol{Q}}^{(e)} \odot \widehat{\boldsymbol{Z}}^{(e)}$
6 $\boldsymbol{R} \leftarrow \mathtt{NTT}^{-1}_{N/2}(\widehat{\boldsymbol{T}}^{(e)})$
7 **return** $\boldsymbol{R}$

In line 3 of Alg. 19, we require an operator which takes in as input a vector of coefficients in base $\mathcal{B}^{(o)}_{\omega,N}$. This vector defines a unique polynomial $\boldsymbol{Q}$ with $\deg(\boldsymbol{Q}) < N/2$. Then this operator must output the vector of coefficients of $\boldsymbol{Q}$ in base $\mathcal{B}^{(e)}_{\omega,N}$. The function BaseConv is

defined for any $\boldsymbol{Q}$ with $\deg(\boldsymbol{Q}) < N/2$ by:

$$\texttt{BaseConv} : (\boldsymbol{Q}(\psi), \boldsymbol{Q}(\psi^3), \ldots, \boldsymbol{Q}(\psi^{N-1})) \mapsto (\boldsymbol{Q}(1), \boldsymbol{Q}(\psi^2), \ldots, \boldsymbol{Q}(\psi^{N-2})) \tag{3.2.3}$$

In [BIN06], (3.2.3) is computed with a classical Lagrange interpolation. Our context is more specific, because the points in which polynomials are evaluated are powers of a N^{th} root of unity ψ in $\mathbb{F}_q$. Therefore it can be implemented with an inverse NTT of size $N/2$ (with ψ^2 as primitive $N/2$-root of unity) with negative-wrapped convolution (nwc) (cf. Section 1.3.2) followed by a classical NTT of size $N/2$. Thus it can be performed basically with one inverse NTT of size $N/2$, $N/2$ modular multiplications and one NTT of size $N/2$.

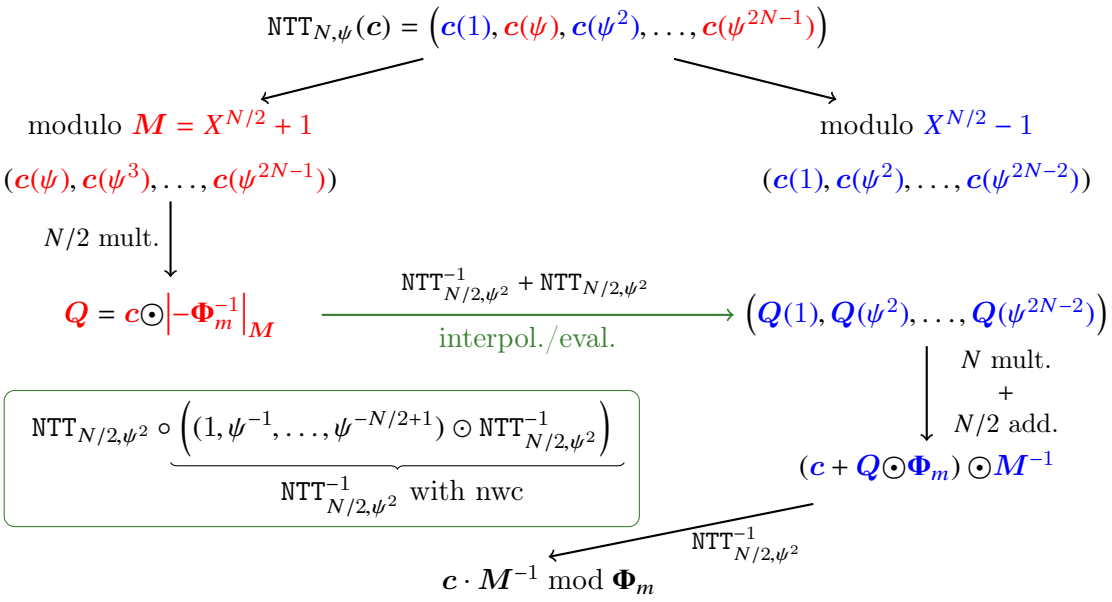


Figure 9: NTT-based Montgomery reduction

Complexity

As a consequence, reducing a polynomial $\boldsymbol{c}$ of degree $2n-2$ with coefficients taken modulo $q = q_1 \cdots q_k$ with Algorithm 19 requires to run it on each RNS channel (modulo each q_i). Therefore its cost in terms of modular multiplications is given by:

$$k \cdot (3 \cdot \mathcal{C}_{\text{NTT}}(\mathcal{N}_2(n)) + 4 \cdot \mathcal{N}_2(n)) .$$

One can find in Table 6 the predicted speed-up of the proposed Montgomery reduction by assuming once again than an NTT of size n requires $n \log_2(n)$ modular multiplications. Despite its lower complexity, the Montgomery algorithm suffers from one main drawback which is the presence of the Montgomery factor in the output.

m	3855	4369	13107	21845	32767	65535
n	2048	4096	8192	16384	27000	32768
Speed-up	2.62	2.62	2.63	2.63	2.63	2.63

Table 6: Theoretical speed-up of Alg. 19 when compared with Alg. 17

3.3 Adaptation of BGV and FV to the Montgomery representation

In this section, we show how the Montgomery representation induced by Algorithm 19 impacts the BGV and FV schemes and suggest modifications to handle these changes. For simplicity, we denote by $\boldsymbol{M}$ the Montgomery factor ($X^{N/2}+1 \bmod \boldsymbol{\Phi}_m$). Thanks to Lemma 3.2.3 we also know that $\boldsymbol{M}^{-1}$ exists in $\mathcal{R}$. We assume that ciphertexts $\widetilde{\mathtt{ct}}$ are given in Montgomery representation, i.e. such that $\widetilde{\mathtt{ct}} = (\widetilde{\boldsymbol{c_0}}, \widetilde{\boldsymbol{c_1}}) = (\boldsymbol{c}_0\boldsymbol{M}, \boldsymbol{c}_1\boldsymbol{M})$. The conversion to the Montgomery domain can be integrated in the encryption procedure for increased efficiency, and the $\boldsymbol{M}$ factor can be removed during decryption by applying a Montgomery reduction to $[\widetilde{\boldsymbol{c_0}} + \widetilde{\boldsymbol{c_1}}\boldsymbol{s}]_q$. The Montgomery reduction only impacts procedures involving multiplications in $\mathcal{R}_q$ by adding a factor $\boldsymbol{M}^{-1}$ to the result. Therefore, homomorphic additions are not affected by this different representation. Notice that the Montgomery representation is stable with respect to multiplications $((\boldsymbol{cM}) \cdot (\boldsymbol{c'M}) = (\boldsymbol{c} \cdot \boldsymbol{c'})\boldsymbol{M}^2$ which is transformed in $(\boldsymbol{c} \cdot \boldsymbol{c'})\boldsymbol{M}$ $(\bmod \boldsymbol{\Phi}_m)$ after the reduction). Thus the only impact one has to consider is on homomorphic multiplication.

3.3.1 Impact of the Montgomery representation in BGV

Homomorphic multiplication in BGV corresponds to a polynomial multiplication of the ciphertexts, therefore when multiplying two ciphertexts, encrypted at the same level j, $\mathtt{ct}_i = (\boldsymbol{c}_{i,0}, \boldsymbol{c}_{i,1}) \in \mathcal{R}_{q_j}^2$ we obtain:

$$\widetilde{\mathtt{ct}}_1 \star \widetilde{\mathtt{ct}}_2 = ((\boldsymbol{c}_{1,0} \cdot \boldsymbol{c}_{2,0}) \cdot \boldsymbol{M}, (\boldsymbol{c}_{1,0} \cdot \boldsymbol{c}_{2,1} + \boldsymbol{c}_{1,1} \cdot \boldsymbol{c}_{2,0}) \cdot \boldsymbol{M}, (\boldsymbol{c}_{1,1} \cdot \boldsymbol{c}_{2,1}) \cdot \boldsymbol{M}) = \widetilde{\mathtt{ct}_1 \star \mathtt{ct}_2} = \widetilde{\mathtt{ct}}_{\text{mult}}$$

Thus, the noise is not affected in this part. As explained in Section 1.2.1 usually a relinearization procedure is applied after each multiplication. Now, we assume that we need to relinearize the ciphertext $\widetilde{\mathtt{ct}}_{mult} = (\widetilde{\boldsymbol{c}}_0, \widetilde{\boldsymbol{c}}_1, \widetilde{\boldsymbol{c}}_2)$. Note that the relinearization procedure of BGV can be adapted to RNS in exactly the same way than for FV (cf. Chapter 2), hence we will adopt the notations of Chapter 2 for this part. Within the RNS variant, the following dot products are computed over $\mathcal{R}_{q_j}$ $\left\langle \mathcal{D}_{\mathtt{RNS},q_j}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_i} \right\rangle$, where $\overrightarrow{\mathtt{rlk}_0} = \left[\mathcal{P}_{\mathtt{RNS},q_L}(\boldsymbol{s}^2) + \vec{\boldsymbol{a}}\,\boldsymbol{s} + t\vec{\boldsymbol{e}}\right]_{q_L}$ and $\overrightarrow{\mathtt{rlk}_1} = \left[-\vec{\boldsymbol{a}}\right]_{q_L}$ (cf. Section 1.2.1). The goal of relinearisation is to obtain $\left\langle \mathcal{D}_{\mathtt{RNS},q_j}(\boldsymbol{c}_2), \mathcal{P}_{\mathtt{RNS},q}(\boldsymbol{s}^2) \right\rangle \equiv$

$\boldsymbol{c}_2\boldsymbol{s}^2 \bmod q_j$ with a limited increase of the noise. Indeed, one can write:

$$\begin{cases} \left\langle \mathcal{D}_{\mathtt{RNS},q_j}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_0} \right\rangle & \equiv & \boldsymbol{c}_2 \cdot \boldsymbol{s}^2 + \left\langle \mathcal{D}_{\mathtt{RNS},q_j}(\boldsymbol{c}_2), \vec{\boldsymbol{a}} \right\rangle \cdot \boldsymbol{s} + t \left\langle \mathcal{D}_{\mathtt{RNS},q_j}(\boldsymbol{c}_2), \vec{\boldsymbol{e}} \right\rangle \bmod q_j \\ & \equiv & \boldsymbol{c}_2\boldsymbol{s}^2 + \boldsymbol{a}'\boldsymbol{s} + t\boldsymbol{e}' \bmod q_j \\ \left\langle \mathcal{D}_{\mathtt{RNS},q_j}(\boldsymbol{c}_2), \overrightarrow{\mathtt{rlk}_1} \right\rangle & \equiv & -\left\langle \mathcal{D}_{\mathtt{RNS},q_j}(\boldsymbol{c}_2), \vec{\boldsymbol{a}} \right\rangle \equiv -\boldsymbol{a}' \bmod q_j \end{cases} \tag{3.3.1}$$

Now, we need to obtain the Montgomery representation of the output of this relinearisation, i.e. a ciphertext like $\left(\left(\boldsymbol{c}_2 \cdot \boldsymbol{s}^2 + \boldsymbol{a}' \cdot \boldsymbol{s} + \boldsymbol{e}'\right) \cdot \boldsymbol{M}, -\boldsymbol{M} \cdot \boldsymbol{a}'\right)$. When the Montgomery representation is used, $\widetilde{\boldsymbol{c}}_2$ replaces $\boldsymbol{c}_2$ in (3.3.1). Therefore if we modify the relinearisation key as follows:

$$\overrightarrow{\mathtt{rlkM}_0} = \left[\left(\mathcal{P}_{RNS,q_L}\left(\boldsymbol{s}^2/\boldsymbol{M}\right) + \vec{\boldsymbol{a}} \cdot \boldsymbol{s} + t\vec{\boldsymbol{e}}\right) \cdot \boldsymbol{M}^2\right]_{q_L},\ \overrightarrow{\mathtt{rlkM}_1} = \left[-\vec{\boldsymbol{a}} \cdot \boldsymbol{M}^2\right]_{q_L} \tag{3.3.2}$$

In the following equations, we simulate the effect of Montgomery reduction by introducing a factor $\boldsymbol{M}^{-1} \pmod{\boldsymbol{\Phi}_m}$:

$$\begin{aligned} \left\langle \mathcal{D}_{\mathtt{RNS},q_j}\left(\widetilde{\boldsymbol{c}}_2\right), \overrightarrow{\mathtt{rlkM}_0} \right\rangle \cdot \boldsymbol{M}^{-1} &\equiv \left[\widetilde{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2 \cdot \boldsymbol{M} + \left\langle \mathcal{D}_{\mathtt{RNS},q_j}\left(\widetilde{\boldsymbol{c}}_2\right), \vec{\boldsymbol{a}} \right\rangle \cdot \boldsymbol{s} \cdot \boldsymbol{M}^2\right] \cdot \boldsymbol{M}^{-1} \\ &\quad + \left\langle \mathcal{D}_{\mathtt{RNS},q_j}\left(\widetilde{\boldsymbol{c}}_2\right), \vec{\boldsymbol{e}} \right\rangle \cdot \boldsymbol{M}^2 \cdot \boldsymbol{M}^{-1} \bmod q_j \\ &\equiv \widetilde{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2 + (\boldsymbol{a}'' \cdot \boldsymbol{s} + \boldsymbol{e}'') \cdot \boldsymbol{M} \bmod q_j \\ &= \left(\boldsymbol{c}_2 \cdot \boldsymbol{s}^2 + \boldsymbol{a}'' \cdot \boldsymbol{s} + \boldsymbol{e}''\right) \cdot \boldsymbol{M} \bmod q_j \end{aligned}$$

Similarly, we get $\left\langle \mathcal{D}_{\mathtt{RNS},q_j}\left(\widetilde{\boldsymbol{c}}_2\right), \overrightarrow{\mathtt{rlkM}_1} \right\rangle \cdot \boldsymbol{M}^{-1} \equiv -\boldsymbol{a}'' \cdot \boldsymbol{M} \bmod q_j$. Hence, we have obtained the Montgomery representation of the output of relinearisation step at no extra cost - both computationally and in terms of noise growth.

Finally, one needs to apply a modulus switching procedure so as to manage noise growth. We consider the ciphertext $(\widetilde{\boldsymbol{c}}_0, \widetilde{\boldsymbol{c}}_1)$ encrypting $\boldsymbol{m}$ at level j and given in Montgomery representation. Let $q_l \mid q_j$ and $\boldsymbol{\delta}_i = [-\widetilde{\boldsymbol{c}}_i/t]_{q_j/q_l} \times t$. Then the modulus switching function applied to $\widetilde{\boldsymbol{c}}_i$ outputs $\widehat{\boldsymbol{c}}_i = (\widetilde{\boldsymbol{c}}_i + \boldsymbol{\delta}_i) \times \frac{q_l}{q_j}$ (cf. Section 1.2.1).

Lemma 3.3.1. If $\left\|[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j}\right\|_\infty < \frac{q_j}{2} - \delta_{\mathcal{R}} \left\|\boldsymbol{M}^{-1}\right\|_\infty \frac{q_j t}{2q_l}\left(1 + \delta_{\mathcal{R}}\|\boldsymbol{s}\|_\infty\right)$ and $q_j = q_l \bmod t$, then

$$\left[(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1}\right]_{q_l} = [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} \bmod t \tag{3.3.3}$$

and

$$\left\|\left[(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1}\right]_{q_l}\right\|_\infty \leqslant \frac{q_l}{q_j}\left\|[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j}\right\|_\infty + t\delta_{\mathcal{R}}\left\|\boldsymbol{M}^{-1}\right\|_\infty \frac{1 + \delta_{\mathcal{R}}\|\boldsymbol{s}\|_\infty}{2} \tag{3.3.4}$$

Proof. It is similar to the proof presented in Section 1.2.1. By definition of $\widetilde{\boldsymbol{c}}_i$, we have:

$$\left[(\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1}\right]_{q_j} = [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} = \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} - q_j\boldsymbol{k}.$$

By definition of $\widehat{\boldsymbol{c}}_i$, we can write:

$$\begin{aligned}(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1} &= \frac{q_l}{q_j}\,(\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s} + \boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1} \\ &= \frac{q_l}{q_j}\,(\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}) + \frac{q_l}{q_j}\,(\boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s})\,\boldsymbol{M}^{-1} \\ &= \frac{q_l}{q_j}\,[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} + q_l\boldsymbol{k} + \frac{q_l}{q_j}\,(\boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1}.\end{aligned} \tag{3.3.5}$$

Moreover, since $\|\boldsymbol{\delta}_i\|_\infty \leqslant \frac{q_j t}{2q_l}$, we get the following bound:

$$\left\|(\boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1}\right\|_\infty \leqslant t\delta_{\mathcal{R}}\frac{q_j}{2q_l}\left\|\boldsymbol{M}^{-1}\right\|_\infty (1 + \delta_{\mathcal{R}}\|\boldsymbol{s}\|_\infty)$$

Thus, from the above and by considering the hypothesis on the norm of $[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j}$, we deduce that:

$$\left\|(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1} - q_l\boldsymbol{k}\right\|_\infty < q_l/2$$

and then that

$$(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1} - q_j\boldsymbol{k} = \left[(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1}\right]_{q_l}.$$

Hence, from this previous equality and by bounding the norm of last member of (3.3.5), we obtain (3.3.4). Finally, we get (3.3.3) by:

$$\begin{aligned}\left[(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1}\right]_{q_l} &= (\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot \boldsymbol{M}^{-1} - q_l\boldsymbol{k} \\ &= (\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\,/\boldsymbol{M} - q\boldsymbol{k} \bmod t \;\; (\widehat{\boldsymbol{c}}_i = \widetilde{\boldsymbol{c}}_i \bmod t; q_j = q_l \bmod t) \\ &= \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} - q_j\boldsymbol{k} \bmod t \text{ (def. of } \widetilde{\boldsymbol{c}}) \\ &= [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_{q_j} \bmod t.\end{aligned}$$

□

From this lemma, we can see that the Montgomery representation of the ciphertext impacts the modulus switching procedure by the addition of an extra factor $\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty$ to the last term on the bound of the hypothesis and of (3.3.4) compared to those given in Section 1.2.1. We will discuss the sizes of $\delta_{\mathcal{R}}$ and $\|\boldsymbol{M}^{-1}\|_\infty$ in the next sections.

3.3.2 Impact of the Montgomery representation in FV

We recall that the first step of the FV homomorphic multiplication corresponds to the extension of ciphertexts to a larger RNS basis, so that coefficients of the product are not reduced modulo q. In order to improve efficiency, an approximate extension is used which introduces an important noise growth which has to be reduced through Algorithm 15, in the end the norm of the polynomials is bounded by $\frac{q}{2}(1+\rho)$ for a parameter $\rho > 0$ (cf. Lemma 2.2.1). Once the ciphertexts to multiply, given in Montgomery representation, have their residues in both bases, the product and the scaling are performed resulting in the degree 2 ciphertext

given below:

$$\widetilde{\mathtt{ct}}_{\text{mult}} = \left(\left[\left\lfloor \frac{t}{q} M \cdot \boldsymbol{c}_0 \cdot \boldsymbol{c}'_0 \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q} M \cdot (\boldsymbol{c}_0 \cdot \boldsymbol{c}'_1 + \boldsymbol{c}_1 \cdot \boldsymbol{c}'_0) \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q} M \cdot \boldsymbol{c}_1 \cdot \boldsymbol{c}'_1 \right\rceil \right]_q \right) \tag{3.3.6}$$

Proposition 3.3.2. Let $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$ and $\mathtt{ct}' = (\boldsymbol{c}'_0, \boldsymbol{c}'_1) \in \mathcal{R}_q^2$ two ciphertexts with inherent noise $\boldsymbol{v}$ and $\boldsymbol{v}'$ respectively and $\widetilde{\mathtt{ct}}_{\text{mult}} = (\widetilde{\boldsymbol{c}}_0, \widetilde{\boldsymbol{c}}_1, \widetilde{\boldsymbol{c}}_2)$ be the degree 2 ciphertext given in equation (3.3.6). By denoting

$$r_\infty^M = \delta_{\mathcal{R}} \|\boldsymbol{M}^{-1}\|_\infty (1+\rho) \frac{\delta_{\mathcal{R}} \|\boldsymbol{s}\|_\infty + 1}{2} + 1$$

we have:

$$(\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s} + \widetilde{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2) \cdot \boldsymbol{M^{-1}} \equiv \Delta \, [\boldsymbol{m}_1 \cdot \boldsymbol{m}_2]_t + \boldsymbol{v}_{\text{mult}} \bmod q$$

with the following bound on the noise:

$$\begin{aligned} \|\boldsymbol{v}_{\text{mult}}\|_\infty \leq & \delta_{\mathcal{R}} t \left(r_\infty^M + \frac{1}{2} \right) (\|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}'\|_\infty) + \frac{\delta_{\mathcal{R}}}{2} \min(\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}\|'_\infty) \\ & + |q|_t \delta_{\mathcal{R}} t \left(r_\infty^M + \frac{\delta_{\mathcal{R}} \|\boldsymbol{M}^{-1}\|_\infty}{2} + 1 \right) + \delta_{\mathcal{R}} \|\boldsymbol{M}^{-1}\|_\infty \frac{1 + \delta_{\mathcal{R}} \|\boldsymbol{s}\|_\infty + \delta_{\mathcal{R}}^2 \|\boldsymbol{s}\|_\infty^2}{2}. \end{aligned} \tag{3.3.7}$$

Proof. First notice that similarly to equation (1.2.16), any valid ciphertext $\widetilde{\mathtt{ct}} = (\widetilde{\boldsymbol{c}}_0, \widetilde{\boldsymbol{c}}_1) \in \mathcal{R}_q^2$ in Montgomery representation satisfies:

$$\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s} = \boldsymbol{M} \cdot (\Delta [\boldsymbol{m}]_t + \boldsymbol{v}) + q\boldsymbol{r} \text{ for some } \boldsymbol{r} \in \mathcal{R}.$$

with:

$$\boldsymbol{r} = \frac{\widetilde{\boldsymbol{c}}_1 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s} - \boldsymbol{M} \cdot (\Delta [\boldsymbol{m}]_t + \boldsymbol{v})}{q}$$

thus:

$$\|\boldsymbol{r} \cdot \boldsymbol{M}^{-1}\|_\infty \leq \delta_{\mathcal{R}} \|\boldsymbol{M}^{-1}\|_\infty (1+\rho) \frac{\delta_{\mathcal{R}} \|\boldsymbol{s}\|_\infty + 1}{2} + 1 = r_\infty^M. \tag{3.3.8}$$

Thus when multiplying two valid ciphertexts $\widetilde{\mathtt{ct}} = (\widetilde{\boldsymbol{c}}_0, \widetilde{\boldsymbol{c}}_1)$ and $\widetilde{\mathtt{ct}'} = (\widetilde{\boldsymbol{c}'}_0, \widetilde{\boldsymbol{c}'}_1)$, in Montgomery representation, encrypted under the key we obtain:

$$\begin{aligned} (\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot (\widetilde{\boldsymbol{c}'}_0 + \widetilde{\boldsymbol{c}'}_1 \cdot \boldsymbol{s}) = & (\Delta \boldsymbol{M} \cdot [\boldsymbol{m}]_t + \boldsymbol{M} \cdot \boldsymbol{v} + q\boldsymbol{r}) \cdot (\Delta \boldsymbol{M} \cdot [\boldsymbol{m}']_t + \boldsymbol{M} \cdot \boldsymbol{v}' + q\boldsymbol{r}') \\ = & \boldsymbol{M} \Delta^2 [\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t + \boldsymbol{M} \Delta ([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + [\boldsymbol{m}']_t \cdot \boldsymbol{v}) \\ & + q\Delta ([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r}) + \boldsymbol{M} \cdot \boldsymbol{v} \cdot \boldsymbol{v}' + q(\boldsymbol{v} \cdot \boldsymbol{r}' + \boldsymbol{v}' \cdot \boldsymbol{r}) \\ & + q^2 \boldsymbol{r} \cdot \boldsymbol{r}' \cdot \boldsymbol{M}^{-1}. \end{aligned}$$

We recall the following useful equalities: $q = \Delta t + |q|_t$, $[\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t = [\boldsymbol{m} \cdot \boldsymbol{m}']_t + t\boldsymbol{r}_m$ and

$\boldsymbol{v} \cdot \boldsymbol{v}' = [\boldsymbol{v} \cdot \boldsymbol{v}']_\Delta + \Delta \boldsymbol{r}_v$. Then by performing the scaling we obtain:

$$\begin{aligned}\frac{t}{q}(\widetilde{\boldsymbol{c}}_1 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s}) \cdot (\widetilde{\boldsymbol{c}'}_1 + \widetilde{\boldsymbol{c}'}_1 \cdot \boldsymbol{s}) =& \boldsymbol{M}\Delta[\boldsymbol{m} \cdot \boldsymbol{m}']_t + \boldsymbol{M} \cdot ([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + [\boldsymbol{m}']_t \cdot \boldsymbol{v}) + tq\boldsymbol{r} \cdot \boldsymbol{r}' \cdot \boldsymbol{M}^{-1} \\ &+ (q - |q|_t)([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r} + \boldsymbol{r}_m) + \boldsymbol{M} \cdot \boldsymbol{r}_v + t(\boldsymbol{v} \cdot \boldsymbol{r}' + \boldsymbol{v}' \cdot \boldsymbol{r}) \\ &+ \boldsymbol{M} \cdot \boldsymbol{r}_r.\end{aligned}$$

with $\boldsymbol{r}_r \in \mathcal{R}$ like in (1.2.18), then by considering the rounding error $\boldsymbol{r}_a \in \mathcal{R}$ (Equation (1.2.19)) we get:

$$\left(\left\lfloor \frac{t}{q}\boldsymbol{M} \cdot \boldsymbol{c}_0 \cdot \boldsymbol{c}'_0 \right\rceil + \left\lfloor \frac{t}{q}\boldsymbol{M} \cdot (\boldsymbol{c}_0 \cdot \boldsymbol{c}'_1 + \boldsymbol{c}'_0 \cdot \boldsymbol{c}_1) \right\rceil \cdot \boldsymbol{s} + \left\lfloor \frac{t}{q}\boldsymbol{M} \cdot \boldsymbol{c}_1 \cdot \boldsymbol{c}'_1 \right\rceil \cdot \boldsymbol{s}^2\right) \cdot \boldsymbol{M}^{-1} \equiv \Delta[\boldsymbol{m} \cdot \boldsymbol{m}']_t + \boldsymbol{v}_{\text{mult}} \ [q]$$

with:

$$\begin{aligned}\boldsymbol{v}_{\text{mult}} =& ([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + [\boldsymbol{m}']_t \cdot \boldsymbol{v}) - |q|_t \boldsymbol{M}^{-1} \cdot ([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r} + \boldsymbol{r}_m) + \boldsymbol{r}_v \\ &+ t\boldsymbol{M}^{-1} \cdot (\boldsymbol{v} \cdot \boldsymbol{r}' + \boldsymbol{v}' \cdot \boldsymbol{r}) + \boldsymbol{r}_r - \boldsymbol{M}^{-1} \cdot \boldsymbol{r}_a.\end{aligned}$$

By using Equations (1.2.6), (3.3.8), (1.2.17), (1.2.18) and (1.2.19) we can bound $\boldsymbol{v}_{\text{mult}}$:

$$\begin{aligned}\|\boldsymbol{v}_{\text{mult}}\|_\infty \leq& \delta_{\mathcal{R}} t \left(r_\infty^{\boldsymbol{M}} + \frac{1}{2}\right)(\|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}'\|_\infty) + \frac{\delta_{\mathcal{R}}}{2}\min(\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}\|'_\infty) \\ &+ |q|_t \delta_{\mathcal{R}} t \left(r_\infty^{\boldsymbol{M}} + \frac{\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty}{2} + 1\right) + \delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty \frac{1 + \delta_{\mathcal{R}}\|\boldsymbol{s}\|_\infty + \delta_{\mathcal{R}}^2\|\boldsymbol{s}\|_\infty^2}{2}.\end{aligned}$$

□

Concerning the relinearization step, an analysis similar to the one in Section 3.3.1 can be performed, with minor adaptations for the relinearisation key. Similarly, one concludes that the Montgomery reduction introduces no extra cost, both in terms of computation and in terms of noise growth, during this step.

3.3.3 Overall impact on noise growth

For both BGV and FV, $\|\boldsymbol{M}^{-1}\|_\infty$ and the expansion factor $\delta_{\mathcal{R}}$ are involved in the noise growth due to the scaling steps performed with the Montgomery reduction.

Therefore in this case, one must take into account the norm of the associated $\boldsymbol{M}^{-1}$ when selecting parameters but this does not seem to be very restrictive. Indeed on the first $20,000$ cyclotomics less than 13.4% have an infinite norm greater than 10, less than 0.5% greater than 100 and only 3 of them greater than $1,000$. Moreover among those cyclotomics, those whose $\boldsymbol{M}^{-1}$'s norm is greater than 100 offer a relatively small number of batching slots compared to their degree. Indeed their ratio $N\log_2(N)/\ell$ is equal to 5343 on average (and lie between 450 and 30720) whereas the cyclotomics proposed in Table 5 have their ratio smaller than 250 (cf. Figure 8). Therefore the cyclotomics whose $\boldsymbol{M}^{-1}$ factor has large coefficients do not seem to be the best suited for batching.

Concerning the expansion factor $\delta_{\mathcal{R}}$, when m is a power of two, it is equal to n but for other m it can be much larger (even super polynomial in m [Erd46]). We can derive an upper bound on the expansion factor for the m-th cyclotomic by considering $\mathcal{F}_m : \mathbb{Q}_{2n-2}[X] \to \mathbb{Q}[X]/(\mathbf{\Phi}_m(X))$, so that $\mathcal{F}_m(\boldsymbol{a}) = \boldsymbol{a} \bmod \mathbf{\Phi}_m$ for every $\boldsymbol{a} \in \mathbb{Q}[X]$ of degree lesser than or equal to $2n-2$.

Lemma 3.3.3. Let m be a positive integer and let $\mathcal{R} = \mathbb{Z}[X]/(\mathbf{\Phi}_m(X))$, with $\deg(\mathbf{\Phi}_m) = n$. If $\delta_{\mathcal{R}}$ denotes the expansion factor of the ring $\mathcal{R}$, then $\delta_{\mathcal{R}} \leq n\|\mathcal{F}_m\|_\infty$.

Proof. Let $\boldsymbol{a}$ and $\boldsymbol{b}$ two elements of $\mathcal{R} - \{0\}$. They naturally embed in $\mathbb{Z}_{n-1}[X] \subset \mathbb{Q}_{2n-2}[X]$. We can write $\|\boldsymbol{ab}\|_\infty \leq n\|\boldsymbol{a}\|_\infty\|\boldsymbol{b}\|_\infty$. As the product $\boldsymbol{ab}$ has degree at most $2n-2$ with coefficients in $\mathbb{Z}$, it belongs to $\mathbb{Q}_{2n-2}[X]$. Since $\mathcal{F}_m$ is a linear map between two vector spaces of finite dimension it is continuous, then we obtain $\|\mathcal{F}_m(\boldsymbol{ab})\|_\infty \leq \|\mathcal{F}_m\|_\infty\|\boldsymbol{ab}\|_\infty \leq n\|\mathcal{F}_m\|_\infty\|\boldsymbol{a}\|_\infty\|\boldsymbol{b}\|_\infty$. □

These two parameters, are given in Table 7 for the different cyclotomic polynomials considered in this chapter. From now, we assume that the key distribution χ_{key} and the error distribution χ_{err} output elements whose infinity norms are bounded by B_{key} and $B_{err} = 6\sigma_{err}$ respectively.

BGV

Similarly to Section 1.2.1, if the two level i ciphertexts have a noise whose norm is bounded by V ($V < \lfloor q_0/2t \rfloor$), we can show that the level $i-1 \geq 0$ ciphertext resulting from the multiplication plus relinearization plus modulus switching procedures has its noise bounded by:

$$\begin{aligned}\|\hat{\boldsymbol{v}}\|_\infty &\leq \frac{q_{i-1}}{q_i}\left(\frac{\delta_{\mathcal{R}}t}{2}\left(V^2+2V+1\right)+\|\boldsymbol{b}_{\text{relin}}\|_\infty\right)+\|\boldsymbol{b}^{\boldsymbol{M}}_{\text{scale}}\|_\infty \\ &\leq \frac{q_{i-1}}{q_i}\frac{\delta_{\mathcal{R}}t}{2}\left(V^2+2V+1\right)+\|\boldsymbol{b}_{\text{relin}}\|_\infty+\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty\|\boldsymbol{b}_{\text{scale}}\|_\infty\end{aligned}$$

where $\boldsymbol{b}_{\text{relin}}$ is the noise caused by the relinearization procedure (cf. Equation (1.2.9)), or its equivalent RNS version (cf. Section 2.3), which remains unchanged with the Montgomery representation, and $\|\boldsymbol{b}^{\boldsymbol{M}}_{\text{scale}}\|_\infty = \delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty\frac{1+\delta_{\mathcal{R}}B_{key}}{2}$ i.e. $\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty\|\boldsymbol{b}_{\text{scale}}\|_\infty$ with $\boldsymbol{b}_{\text{scale}}$ the noise caused by the scaling in the original setting (cf. Equation (1.2.4)). Therefore if the parameters are selected such that:

$$\begin{cases} \dfrac{q_i}{q_{i-1}} \geq \delta_{\mathcal{R}}tV \text{ for all } 1 \leq i \leq L \\ 2+\|\boldsymbol{b}_{\text{relin}}\|_\infty+\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty\|\boldsymbol{b}_{\text{scale}}\|_\infty \leq \dfrac{V}{2} \end{cases} \tag{3.3.9}$$

we obtain:

$$\begin{aligned}\|\hat{v}\|_\infty &\leq \frac{1}{2V}\left(V^2+2V+1\right)+\|\boldsymbol{b}_{\text{relin}}\|_\infty+\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty\|\boldsymbol{b}_{\text{scale}}\|_\infty\\ &\leq \frac{V}{2}+2+\|\boldsymbol{b}_{\text{relin}}\|_\infty+\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty\|\boldsymbol{b}_{\text{scale}}\|_\infty\\ &\leq V\end{aligned}$$

which means that for a chain of $L+1$ moduli, we are able to evaluate circuits of multiplicative depth L. Since the factor $\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty$ is only added to the noise in the end, the impact of the Montgomery representation on the parameter selection is quite moderate, actualy in practice the parameters can remain the same most of the time.

FV

The initial noise of a ciphertext is at most $V_{init} = B_{err}(1+2\delta_{\mathcal{R}}B_{key})$ (cf. Section 1.2.2). We recall that $r_\infty^{\boldsymbol{M}} = \delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty(1+\rho)\frac{\delta_{\mathcal{R}}B_{key}+1}{2}+1$ and that the k moduli q_i have size ν i.e. $q_i < 2^\nu$. The output of a tree of depth L has a noise bounded by $C^L_{\text{RNS},\boldsymbol{M},1}V + C_{\text{RNS},\boldsymbol{M},2}\frac{C^L_{\text{RNS},\boldsymbol{M},1}-1}{C_{\text{RNS},\boldsymbol{M},1}-1}$ (cf. Section 1.2.2) with:

$$\begin{cases} C_{\text{RNS},\boldsymbol{M},1} &= \delta_{\mathcal{R}}t\left(2r_\infty^{\boldsymbol{M}}+1\right)+\dfrac{\delta_{\mathcal{R}}}{2}\\ C_{\text{RNS},\boldsymbol{M},2} &= |q|_t\delta_{\mathcal{R}}t\left(r_\infty^{\boldsymbol{M}}+\dfrac{\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty}{2}+1\right)+\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty\dfrac{1+\delta_{\mathcal{R}}B_{key}+\delta_{\mathcal{R}}^2B_{key}^2}{2}\\ &\quad +k(1+\delta_{\mathcal{R}}B_{key}+\delta_{\mathcal{R}}^2B_{key}^2)+k\delta_{\mathcal{R}}B_{err}2^{\nu+1}.\end{cases} \tag{3.3.10}$$

We denote by $L^{\boldsymbol{M}}_{max} = \max\left\{L\in\mathbb{N} \mid C^L_{\text{RNS},\boldsymbol{M},1}V + C_{\text{RNS},\boldsymbol{M},2}\frac{C^L_{\text{RNS},\boldsymbol{M},1}-1}{C_{\text{RNS},\boldsymbol{M},1}-1} \leqslant B_{dec}\right\}$ the depth allowed by the homomorphic multiplication where B_{dec} corresponds to the decryption bound of the RNS variant of `FV` given in (2.1.4). Since a factor $\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty$ is roughly added to the orginal $C_{\text{RNS},1}$ constant (cf. Equation (2.4.1)), we can expect an important impact on the multiplicative depth.

Table 7 presents the maximal theoretical depths for `FV` with and without the use of the Montgomery reduction ($L^{\boldsymbol{M}}_{max}$ and L_{max} respectively). For these computations we have taken parameters $B_{key} = 1$, following the recommendations of [BF16] $\sigma_{err} = 2\sqrt{n}$, and a number k of 62-bits moduli to get the largest size for q ensuring at least 80-bits of security according to [APS15]. We notice that, as expected, the multiplicative depth is far smaller with a Montgomery representation in theory and we have been able to confirm this behaviour in practice.

Mixing optimized Barrett and Montgomery reductions Considering the non-negligible impact of the Montgomery representation on the multiplicative depth of `FV`, a more robust strategy for this cryptosystem corresponds to a mixed Barrett/Montgomery approach. Alg.

m	n	k	$\|\|\boldsymbol{M}^{-1}\|\|_\infty$	$\delta_{\mathcal{R}}$	L_{max}	L^{M}_{max}
4369	4096	2	1	$35n$	1 (4)	1 (3)
13107	8192	5	1	$205n$	6 (17)	4 (10)
21845	16384	11	1	$739n$	13 (40)	9 (22)
32767	27000	18	9	$2621n$	19 (66)	12 (39)
65535	32768	22	1	$9886n$	22 (80)	15 (45)

Table 7: Theoretical depths with and without Montgomery reduction. Values in parenthesis are the depths observed in practice.

18 is used during the first stage of homomorphic multiplication, with ciphertexts not exploiting a Montgomery representation. This avoids the noise growth caused by the Montgomery factor. Nonetheless, the Montgomery reduction can still be used during the relinearisation stage, since we have seen that this does not cause a larger noise growth. Since we do not want to obtain a ciphertext in Montgomery representation after the relinearization procedure, the relinearisation key needs to be modified by replacing the factor $\boldsymbol{M}^2$ of the Montgomery approach (cf. Equation 3.3.2) by $\boldsymbol{M}$.

3.3.4 About the expansion factor

The reader may have noticed the important difference between the theoretical and practical depth for `FV` presented in Table 7. Of course, the theoretical depth as it is computed represents the worst-case scenario, thus we expect it to be smaller than the depth observed in practice on which we have no guarantee. However for practical applications it could be interesting to have a better approximation of the noise growth behaviour. By definition the expansion factor $\delta_{\mathcal{R}}$ quantifies the growth of the coefficients after a product in the worst-case scenario, thus this is the quantity we should try to evaluate more precisely in practice.

Let us assume we want to multiply two polynomials $\boldsymbol{a}$ and $\boldsymbol{b}$, both of degree $n-1$. Without reducing the product by $\boldsymbol{\Phi}_m$, we have the following bound:

$$\|\boldsymbol{a}\cdot\boldsymbol{b}\|_\infty \leq n\|\boldsymbol{a}\|_\infty\|\boldsymbol{b}\|_\infty$$

For this bound to be reached in practice it is necessary to have all the coefficients of $\boldsymbol{a}$ (resp. $\boldsymbol{b}$) being equals to $\|\boldsymbol{a}\|_\infty$ (resp. $\|\boldsymbol{b}\|_\infty$) at the same time. If we consider that the coefficients of $\boldsymbol{a}$ and $\boldsymbol{b}$ to be sampled independently from a probabilistic distribution, this bound is likely to be reached with exponentially low probability in n. From now we assume our two polynomials have coefficients $(a_0,\ldots,a_{n-1})$ and $(b_0,\ldots,b_{n-1})$ sampled independently and uniformly in a centered interval $[-\mu,\mu]$ whose size is exponential in n. To estimate $\delta_{\mathcal{R}}$, we need to estimate the distribution of:

$$\delta'_i = \sum_{j=0}^{i}\frac{a_j b_{i-j}}{\|\boldsymbol{a}\|_\infty\|\boldsymbol{b}\|_\infty} = \sum_{j=0}^{i}\frac{a_j}{\|\boldsymbol{a}\|_\infty}\frac{b_{i-j}}{\|\boldsymbol{b}\|_\infty}$$

for $i\in\{0,\ldots,2n-2\}$. Now if we assume than $(a_0/\|\boldsymbol{a}\|_\infty,\ldots,a_{n-1}/\|\boldsymbol{a}\|_\infty)$ has the same distribution than $(T_i^{\boldsymbol{a}})_{i=0}^{n-1}$ a sequence of independent random variables uniform in $[-1,1]$. Of course this is not true since the $a_i/\|\boldsymbol{a}\|_\infty$ have only rational values and at least one of them is equal to

± 1 with probability 1/2, but we assume this is a good enough approximation for our purpose.

Therefore δ'_i has the same distribution than $\sum_{j=0}^{i} T_j^{\boldsymbol{a}} T_{i-j}^{\boldsymbol{b}}$. Notice that for $j = 0, \dots i$, the random variables $Z_j^i = T_j^{\boldsymbol{a}} T_{i-j}^{\boldsymbol{b}}$ are independents and have the same probability distribution. We recall that if $Z = XY$ with X and Y two independent random variables following a uniform distribution in $[-\mu, \mu]$, then:

$$\mathbb{P}(Z \leq x) = \frac{1}{2} + \frac{x}{2\mu^2}\left(1 + \ln\left(\frac{\mu^2}{|x|}\right)\right) \text{ for } x \in [-\mu^2, \mu^2] \quad \mathrm{E}(Z) = 0 \quad \mathrm{Var}(Z) = \sigma_Z^2 = \mu^2/9$$

Since δ'_i, $0 \leq i \leq 2n-2$, is a sum of $\alpha_i = \min(i+1, 2n-i-1)$ i.i.d. random variables, the central limit theorem tells us it has asymptotically the same distribution than $\mathcal{N}(0, \alpha_i/9)$.

Now, if we assume the δ'_is to be independents, since the reduction modulo $\boldsymbol{\Phi}_m$ is a linear transformation we can deduce the probabilistic law followed by the coefficients of $\boldsymbol{a} \cdot \boldsymbol{b}/(\|\boldsymbol{a}\|_\infty \|\boldsymbol{b}\|_\infty) \bmod \boldsymbol{\Phi}_m$. Indeed, if X and Y are two independent random variables such that $X \rightsquigarrow \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y \rightsquigarrow \mathcal{N}(\mu_2, \sigma_2^2)$ then $X+Y \rightsquigarrow \mathcal{N}(\mu_1+\mu_2, \sigma_1^2+\sigma_2^2)$ and for any $\lambda \in \mathbb{R}$, $\lambda X \rightsquigarrow \mathcal{N}(\lambda\mu_1, \lambda^2\sigma_1^2)$. We denote $\delta = (\delta_0, \dots, \delta_{n-1}) = \mathcal{F}_m(\delta'_0, \dots, \delta'_{2n-2})$ where $\mathcal{F}_m$ denotes the reduction modulo $\boldsymbol{\Phi}_m$ as in Lemma 3.3.3.

All the coefficients of δ follow a centered gaussian distribution for which we can compute the standard deviation σ_i for each $i = 0, \dots, n-1$, let $\sigma_{\max}$ be the biggest one, thus $|\delta_i|$ follows a half-normal distribution of expected value $\sqrt{2/\pi}\sigma_i$. We know that $\mathbb{P}(|\delta_i| > x\sigma_i) = \texttt{erfc}(x/\sqrt{2})$ where erfc denotes the complementary error function, as a consequence we can write the following:

$$\mathbb{P}(\|\boldsymbol{a} \cdot \boldsymbol{b}\|_\infty \geq x\sigma_{\max}\|\boldsymbol{a}\|_\infty \|\boldsymbol{b}\|_\infty) \leq n \times \texttt{erfc}\left(x/\sqrt{2}\right)$$

Hence, for a small enough probability (smaller than 2^{-p}), i.e. for a large enough x, we can consider $\delta_p = x\sigma_{max}$ as a probabilistic bound on the expansion factor. The following table presents the values of $\sigma_{\max}$ obtained for the different cyclotomics considered in this chapter together with the bound δ_p for $p = 50$. Finally we indicate in the Table below the multiplicative depths obtained for `FV` with these bounds. It appears that these bounds are still around twice smaller than what is observed in practice, however they are twice larger than the worst-case bounds. Of course to get these bounds we were forced to make

m	$\sigma_{\max}$	δ_{50}	L_{max}	L_{max}^M
4369	122.32	1114	2	2
13107	410.32	3734	9	7
21845	1313.88	12088	21	15
32767	3471.06	32281	33	21
65535	7004.65	65143	39	26

Table 8: Bounds, with faillure probability smaller than 2^{-50}, on the expansion factor for different cyclotomic polynomials with the associated multiplicative depths for `FV`.

several approximations (distributions of the $a_i/\|\boldsymbol{a}\|_\infty$, independence of the δ'_is), thus one could legitimate wonder how reliable our results are. We have run some experiments to check whether our model was far from the reality or not.

We present hereafter the results of these experiments for two of the considered cyclotomics. The results were measured by performing 2^{13} products of polynomials whose coefficients were sampled uniformly in $[-2^{-25}, 2^{25}]$ by using the `rand` function of the C standard library. The left part of the folllowing graphics presents the values observed on our experiments for each coefficient (maximal and average) while the right part presents the values given by our model (expected values and $2\sigma_i$).

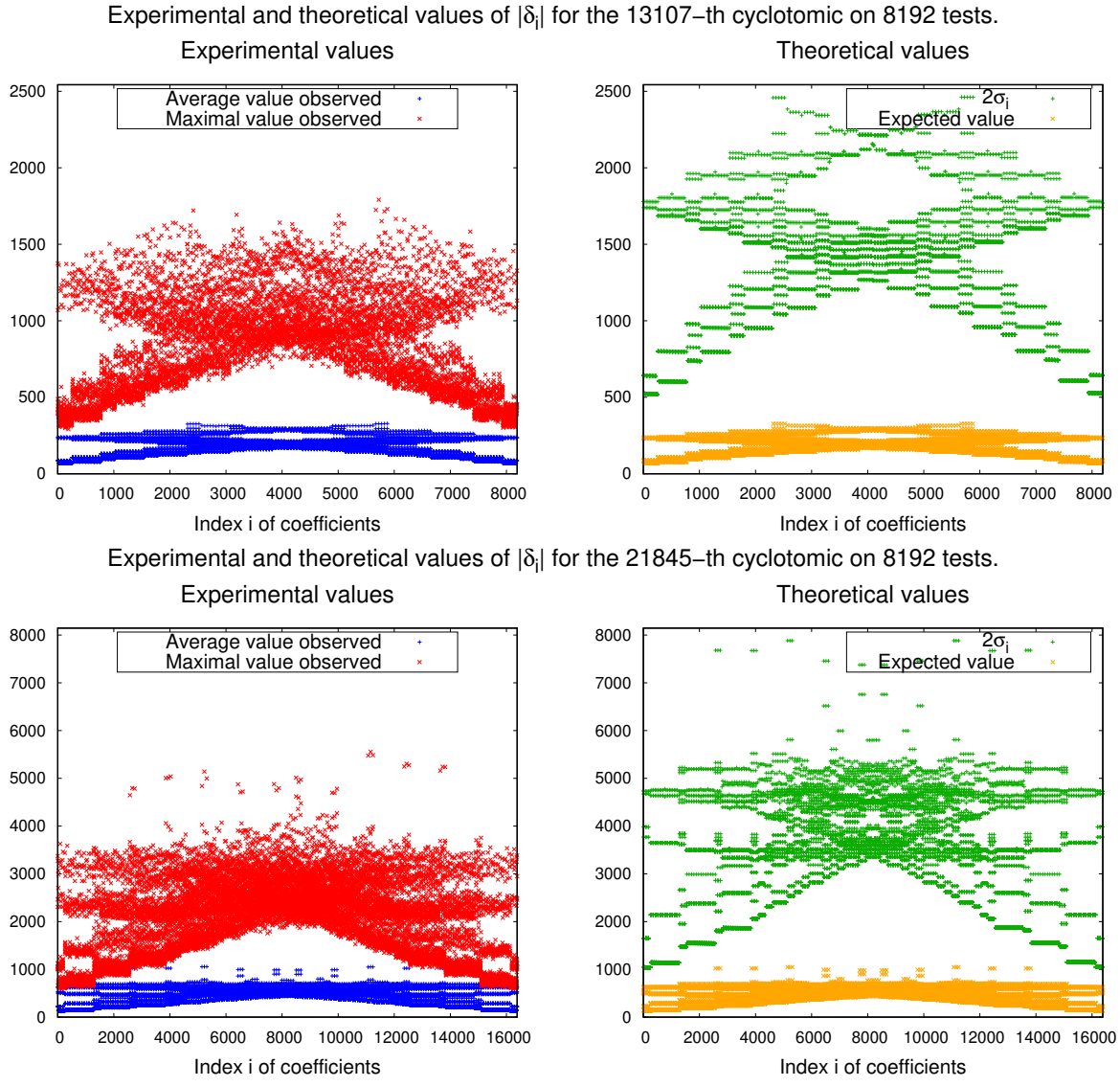


Table 9 presents some indicators of the differences between the model and our experiments. It contains the average (resp. the biggest) relative error $\overline{\varepsilon}_a$ (resp. $\varepsilon_a^{\max}$) between the expected value $\sqrt{2/\pi}\sigma_i$ and the values observed in average. It also indicates the average relative error $\overline{\varepsilon}_m$ (resp. the smallest relative error $\varepsilon_m^{\min}$) between, $2\sigma_i$ and the maximal value observed.

Despite our gross approximations, the results are quite accurate: the expected and average values differ from less than 0.7% in average and less than 3.3% in the worst case, moreover all the values obtained were smaller than $2\sigma_{\max}$ (around 33% in average and 5% in the closest

m	$\overline{\varepsilon}_a$	$\varepsilon_a^{\max}$	$\overline{\varepsilon}_m$	$\varepsilon_m^{\min}$
4369	0.0050	0.025	0.337	0.067
13107	0.0056	0.025	0.328	0.093
21845	0.0063	0.032	0.343	0.102
65535	0.0061	0.032	0.332	0.052

Table 9: Relative errors between the theoretical values and the one observed in practice (in average and in the worst case) for different cyclotomic polynomials.

case). Although further statistical analysis of these results would be required, the first results tend to indicate the model is reasonable enough for our purpose.

3.4 Experimental results

The proposed methods for polynomial reduction have been implemented using C++, and compiled with `gcc` using the optimization flag `-O3`. All the experimental results presented herein were measured on an i7-5960X, running at 3.0 GHz with 32 GB of main memory without exploiting any parallelism.

3.4.1 Polynomial reduction

In Figure 10, one can find the execution timings of polynomial reduction, the unoptimized and optimized Barrett reductions and the Montgomery reduction for the different cyclotomics we have considered. As a general indication, and to highlight the benefit of our reduction algorithms, Figure 10 also presents the timings of a generic reduction algorithm, in our case we have chosen NTL's one by using preconditioning ([HHSSD17]). All timings correspond to the execution of the algorithms on a single modulus of 62-bits. Speed-ups up to 1.95 and 2.55 were achieved for the optimized Barrett and Montgomery algorithms respectively when compared with the unoptimized Barrett reduction. It should be noted that even though the execution times increase with m, allowing to use FHE parameters with a larger multiplicative depth, the time per batching slot presents very little variations for the considered polynomials. Details timings and speed-ups are given in Table 10, one can notice that we obtain speed-ups comparable to what we expected (cf. Tables 5 and 6).

m	3,855	4,369	13,107	21,845	32,767	65,535
Alg. 17	0.351	0.763	1.660	3.543	7.907	7.941
Alg. 18	0.183 (1.92)	0.392 (1.95)	0.927 (1.79)	1.967 (1.8)	5.189 (1.52)	4.976 (1.6)
Alg. 19	0.138 (2.54)	0.301 (2.53)	0.650 (2.55)	1.402 (2.53)	3.167 (2.5)	3.178 (2.5)

Table 10: Timings (ms) of our different reduction algorithms for the m-th cyclotomic polynomial on a single 62-bit modulus. Values in parenthesis are the speed-up when compared to Alg. 17.

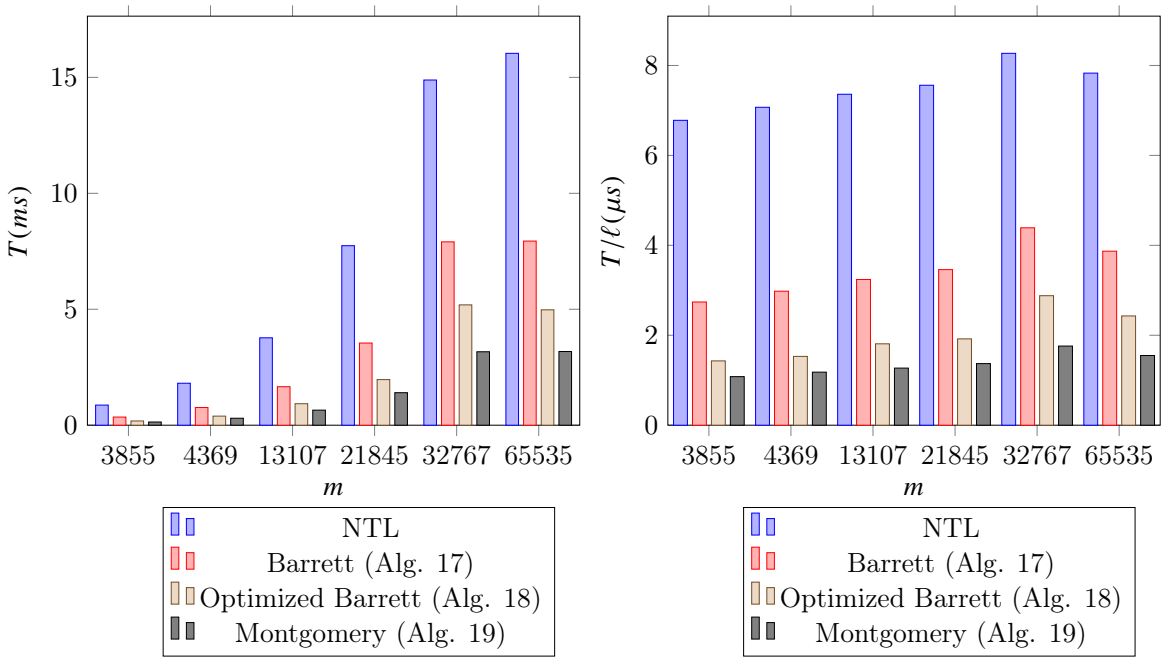


Figure 10: Execution time T (ms) (resp. per batching slot T/ℓ (μs)) for multiple reduction strategies for cyclotomic polynomials of index m on a single 62-bit modulus.

3.4.2 Impact on homomorphic multiplication

The aforementioned reduction methods were used to implement the homomorphic multiplication operations of the `FV` and `BGV` schemes. One can find in Figures 11 and 12 the execution times (resp. execution times divided by the number of batching slots) of the homomorphic multiplication of two freshly encrypted ciphertexts for `FV` and `BGV`, with the parameters given in Table 7. Because of their low performances, the results using NTL reduction are omitted. In order to compare with the most efficient case, we provide instead timings for power-of-two cyclotomics of same dimension using the `NTT`s with negative-wrapped convolution (NWC) algorithm from NFLlib ([AMBG+16]). For instance we compare $m = 21,845$ of degree $n = 2^{14}$ with $m = 2^{15}$ of same degree. Unlike Figure 10, the execution time of homomorphic multiplication increases significantly with m. This trend is explained by the relinearisation procedure, which requires a number of `NTT`s that increases quadratically with $\log_2 q$ (cf. Section 2.3.4).

Nevertheless, the employed reduction procedure plays a preponderant role in the efficiency of the homomorphic multiplication. Indeed, one achieves speed-ups up from 1.24 to 1.12 (resp. from 1.37 to 1.19) when comparing the homomorphic multiplication exploiting the optimized and unoptimized Barrett reduction methods for the `BGV` (resp. `FV`) scheme. Since using only Montgomery reduction in `FV` causes much more important noise growth (cf. Section 3.3.3), we provide the timings corresponding to the mixed Barrett/Montgomery stategy (cf. Section 3.3.3), which does not impact the noise growth, and gave speed-ups from 1.39 to 1.19. In contrast, for `BGV`, where one can exploit the Montgomery representation throughout the whole procedure without impacting significantly the noise growth, we obtain speed-ups from 1.18 to 1.33. Detailed timings can be found in Table 11.

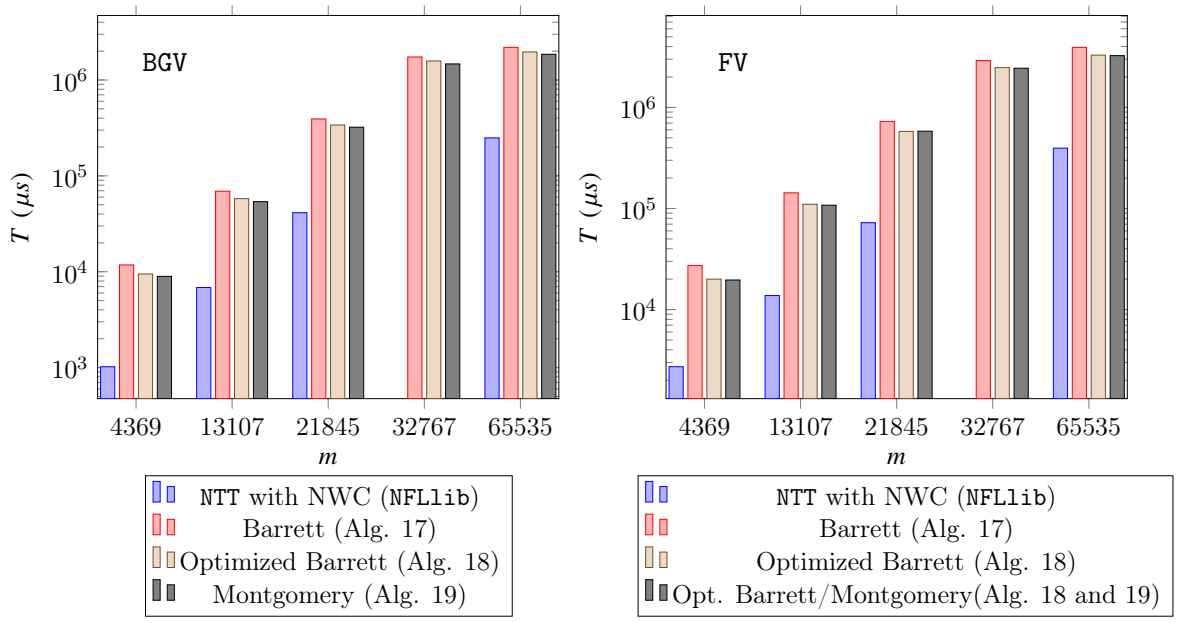


Figure 11: Execution time T (μs) for the multiplication plus relinearization of BGV and FV with several reduction strategies (plus modulus switching for BGV) on the m-th cyclotomic polynomial. Modulus q is a product of 62-bit prime moduli and y-axis is in logarithmic scale.

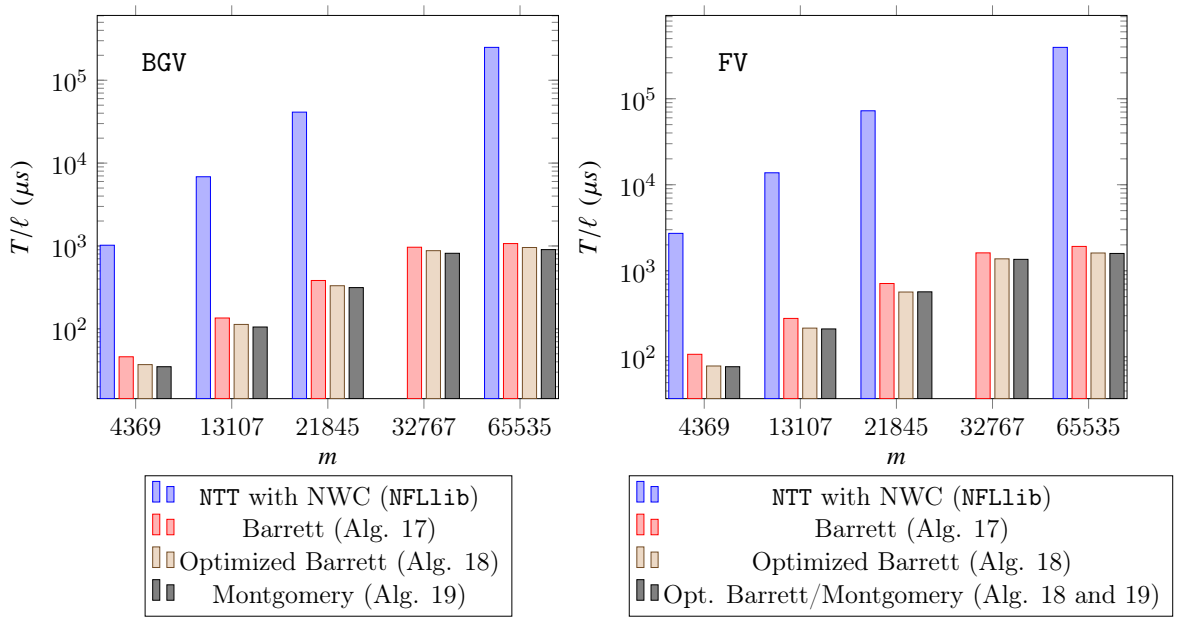


Figure 12: Execution time per slot T/ℓ (μs) for the multiplication plus relinearization of BGV and FV with several reduction strategies (plus modulus switching for BGV) on the m-th cyclotomic polynomial. Modulus q is a product of 62-bit prime moduli and the y-axis is in logarithmic scale

Last, one can notice that using power-of-two cyclotomics with NTTs and negative-wrapped convolution brings a speed-up of roughly 10 , for both BGV and FV, compared to the cyclotomic we have chosen. However when considering the time spent per batchig slots, non-power-of-two cyclotomics are clearly more interesting, with a speed-ups up to 100.

The reader may have noticed that the speed-up of the proposed methods decreases as the

degree n of the cyclotomic polynomial, and thus when $\log_2 q$ gets larger due to the increasing complexity of the relinearisation procedure. This suggests that they are most beneficial when one needs to homomorphically evaluate small circuits. Since a lot of practical applications of FHE ([GDL+16], [KGV16], [NLV11]) have circuits with small depth, the reduction algorithms we have proposed can potentially have a wide range of applicability.

m	$\log_2 q$	BGV			FV		
		Alg. 17	Alg. 18	Alg. 19	Alg. 17	Alg. 18	Mix Alg. 18/19
4,369	124	11.8	9.5 (1.24)	8.9 (1.33)	27.3	20 (1.37)	19.6 (1.39)
13,107	310	69.2	57.8 (1.2)	53.8 (1.29)	142.9	110 (1.3)	107.8 (1.33)
21,845	682	392.5	339.2 (1.16)	322 (1.22)	727.7	579.2 (1.26)	582.1 (1.25)
32,767	1,116	1,738	1,580 (1.1)	1,472 (1.18)	2,905	2,476 (1.17)	2,442 (1.19)
65,535	1,364	2,190	1,963 (1.12)	1,855 (1.18)	3,928	3,294 (1.19)	3,255 (1.21)

Table 11: Timings (ms) of homomorphic multiplication plus relinearization for `BGV` and `FV` encryption schemes (plus modulus switching for `BGV`) on the m-th cyclotomic polynomial. Values in parenthesis correspond to the speed-up compared to Alg. 17.

3.5 Conclusion

Properties of non-power of two cyclotomics allow to pack several binary plaintexts in a single ciphertext and thus to amortize the important costs required by homomorphic encryption. Although Lyubashevsky et al. ([LPR13]) have shown one could generalize the efficient NTT with negative wrapped convolution technique to general cyclotomics, the tensored representation required for their method is less convenient to implement. As a consequence, their method remains little used in practice and major homomorphic libraries such as `HElib` keep using the simpler univariate representation of elements in the general case.

In this work, we have considered and improved the arithmetic associated to the univariate representation in general cyclotomic rings. We have proposed two methods for performing the polynomial reduction: one based on the Barrett reduction and the other on a Montgomery representation both with the same asymptotic complexity. The latter offers better performances, however it causes more important noise growth in homomorphic schemes. If this growth remains small enough to not affect the multiplicative depth of `BGV`, it reduces considerably the practical and theoretical one of scale-invariant schemes like `FV`.

We have highlighted the gains brought by our reduction algorithms in practice with a C++ implementation and have obseverved speed-ups up to 1.95 and 2.55 when comparing, respectively, our optimized Barrett and Montgomery reductions with a classical Barrett reduction. Finally, these reductions have been incorporated into the homomorphic multiplication procedures of `BGV` and `FV`, resulting in speed-ups up to 1.33 and 1.49 respectively. If our optimzed Barrett reduction remains hardly compatible with generic implementation, our Montgomery reduction however is completely generic and could benefit to implementation of homomorphic libraries. In particular, since the noise growth caused by the Montgomery representation does not impact significantly the choice of parameters of `BGV`, `HElib` could entirely benefit of this

reduction procedure.

Part of this work was presented in the 24$^{\text{th}}$ Selected Areas of Cryptography (SAC) conference ([BEH+18]).

Chapter 4

Homomorphic evaluation of support vector machine

Contents

As the world moves into an ever more digital-based economy, those who hold the best quality data have a significant advantage over their competitors. Companies can identify patterns in their users' behaviours, and extrapolate their needs of tomorrow [QWD+16]. As they gather more data, their predictions get increasingly better, further consolidating their position as market leaders. We can foresee how this strategy can be applied to other markets. For instance, a pharmaceutical company, who has collected data about the reactions of a large number of patients to a particular medicine, could potentially extrapolate how other patients would react to it.

Companies can build high-quality models to predict future patterns based on the large amounts of data they collect. When these predictions are made as part of a service, for instance which are the most relevant pages based on a user's query, or which are the products a user is most likely to buy based on his past choices, companies will be invested in preventing the model parameters from being disclosed to a competitor that could potentially mimic their service. In addition, the clients of this service may wish to protect their own data. For example, when doctors want to know how patients would react to a certain medicine, the patients' data should remain confidential.

In its most basic form, one model would produce a single bit based on the input, signaling, for example, whether a certain medicine will be effective based on the patient's heart pressure, age, etc. A SVM is a machine learning method that works through analogy [CL11]. When a model is built, a subset of the examples is encoded as support-vectors, along with weights. During classification, the similarity between the input vector and the examples is evaluated through a kernel function, and the resulting values are combined through a weighted average. The sign of the weighted average identifies whether the input belongs to a certain class or not. In a confidential setting, a user should be able to provide encrypted data to be classified, and a service provider, namely a company, should be able to return a single encrypted bit, encoding the classification result; without the client learning anything else about the model, and without the server learning anything about the input.

Homomorphic encryption offers exactly these properties and is thus perfectly suited to address this problem. However performing efficiently the computations required by the SVM from the homomorphic additions and multiplications primitives is quiet challenging. In this chapter we present efficient methods to perform the computations required by the SVMs quite efficiently. For this purpose, we use the construction of Cheon et al. ([CKKS17]) which allows to perform approximate arithmetic homomorphically.

4.1 Preliminaries

This section starts by giving an overview of the related art and in particular of previous works using homomorphic encryption for the classification of data. Then we briefly present the different computations that need to be done for evaluating SVM and the construction of Cheon et al. that we are using.

4.1.1 Related Art

Homomorphic encryption has been typically exploited to implement either bitwise circuits ([GHS12b], [MS17]) or operations over the integers ([CSVW17]). SVMs are challenging to implement in this setting since they mix both integer arithmetic (computation of the kernel functions) and bitwise circuits (extraction of the sign bit). In the past, this latter problem has been solved by using Garbled Circuits (GCs), requiring interactive protocols ([LLM06]).

In contrast to methods based on GCs, approaches based on homomorphic encryption to classify private data, after transmitting key material, require no communication other than the encrypted data to be classified, achieving the minimal possible overhead. Previous works on the applications of homomorphic encryption to machine learning has mostly considered the use of exact arithmetic ([GLN13], [BCIV17], [CGH+18], [BV18]). However in [KSW+18], Kim et al. have shown that the use of an approximate arithmetic does not affect significantly the quality of the result but has the potential to considerably improve the performance.

The previous works on the application of Homomorphic Encryption (HE) to machine learning have either considered that the computation of threshold functions could be deferred until after the decryption ([GLN13]), or focused on systems, such as logistic regression, where no threshold computation is required ([KSW+18], [CGH+18], [BV18]). Difficulties to implement the sign function come from the fact that the operations available through homomorphic encryption are more compatible with the computation of circuits which can be approximated with polynomials.

However, providing the client with a value rather than its sign may lead to a larger leak of information than a service provider would be willing to accept. Nevertheless, recent research has shown inceptive results pertaining to the computation of non-trivial functions, including the threshold ([CDSM15]) or reduction modulo an integer ([CHK+18]), through an approximation using sine waves. In this work we use a different method, based on the Newton-Raphson procedure, which converges quadratically to the sign function; and apply it to the computation of SVM classification.

4.1.2 Support Vector Machine

Given a multiset of examples $\{(\boldsymbol{x}_i, y_i)\}_i$, each encoded as a vector of features $\boldsymbol{x}_i \in \mathbb{R}^h$ with the corresponding labels $y_i \in \{-1, 1\}$, an SVM selects a subset of them of size r as support-vectors, assigns them weights α_i and a bias β, and computes the parameters to a kernel function K that maximise the accuracy of the resulting model. This corresponds to the training phase. During the classification phase, the model is provided with a vector of features $\boldsymbol{x}$ and will output the most likely label y. The kernel function is used to compute the similarity between the support-vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}$, and the similarity levels are aggregated through a biased weighted average to output the most likely label x belongs to:

$$y = \text{sign}\left(\sum_{i=0}^{r} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + \beta\right) \tag{4.1.1}$$

The four most commonly used kernel functions are:

$$K_{\text{linear}}(\boldsymbol{x}, \boldsymbol{x}_i) = \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle \tag{4.1.2}$$

$$K^{\gamma,\rho,d}_{\text{polynomial}}(\boldsymbol{x}, \boldsymbol{x}_i) = (\gamma \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle + \rho)^d \tag{4.1.3}$$

$$K^{\gamma,\rho}_{\text{sigmoid}}(\boldsymbol{x}, \boldsymbol{x}_i) = \tanh(\gamma \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle + \rho) \tag{4.1.4}$$

$$K^{\gamma}_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}_i) = e^{-\gamma ||\boldsymbol{x} - \boldsymbol{x}_i||^2} \tag{4.1.5}$$

While the first is the most computationally efficient, it is mostly useful for data with a large number of features. In contrast, kernels like $K_{\text{polynomial}}$, K_{RBF} and K_{sigmoid} are best at generalising patterns from data with fewer features.

4.1.3 Homomorphic encryption for approximate numbers

We recall that the m^{th} cyclotomic field $\mathbb{Q}(\zeta_m)$ can be embedded in the subring $H = \{(z_1, \ldots, z_n) : z_{n/2+j} = \overline{z}_j, 1 \leqslant j \leqslant n/2\}$ of $\mathbb{C}^n$ (Equation 1.1.2) through the canonical embedding. Instead of trying to encode data in the coefficients of a polynomial $\boldsymbol{m} \in \mathcal{R}_t$ for some plaintext modulus t as it was usually done, Cheon et al have chosen to encode the plaintext data directly in H. In this way, they can encode directly $n/2$ complex values as an element $\boldsymbol{z}$ of H and thus as an element $\boldsymbol{m}$ of $\mathcal{R}$.

However since $\sigma(\mathcal{R}) \subsetneq H$, one first needs to round $\boldsymbol{z}$ to an element of $\sigma(\mathcal{R})$. There are different methods to perform this kind of rounding in general cyclotomic ring, see for instance [LPR13]. However in the particular case of power-of-two cyclotomics the canonical embedding σ is an isometry, which means in particular that the columns of the matrix $\texttt{CRT}_m$ (Equation (1.3.13)) form an orthogonal basis of the lattice $\sigma(\mathcal{R})$. Therefore, in this case it is staightforward to compute the closest point of $\boldsymbol{z} \in H$ belonging to the lattice $\sigma(\mathcal{R})$ with orthogonal projections:

$$\lfloor \boldsymbol{z} \rceil_{\sigma(\mathcal{R})} = \boldsymbol{z} + \sum_{i=0}^{n-1} \left\lfloor \frac{\left\langle -\boldsymbol{z}, \overline{\sigma(X^i)} \right\rangle}{\|\sigma(X^i)\|_2^2} \right\rceil_1 \sigma(X^i) = \boldsymbol{z} + \boldsymbol{f} \tag{4.1.6}$$

where $\lfloor \cdot \rceil_1$ denotes the fractional part between $[-1/2, 1/2)$ of a real number, we extend this notations to the complex numbers by considering it both on the real and imaginary parts. Once $\boldsymbol{z}$ approximated by its closest point in the lattice, one can recover $\boldsymbol{m} = \sigma^{-1}(\lfloor z \rceil_{\sigma(\mathcal{R})}) \in \mathcal{R}$. Actually since it is an isometry, it is equivalent to directly round the result of the inversion coefficient-wise, i.e. to compute $\boldsymbol{m} = \lfloor \sigma^{-1}(z) \rceil$. Finally, in order to avoid the rounding to affect the most significant bits of $\boldsymbol{z}$, one should multiply $\boldsymbol{z}$ by a large factor Δ before performing the rounding, so that in the end $\boldsymbol{m} = \sigma^{-1}(\lfloor \Delta \boldsymbol{z} \rceil_{\sigma(\mathcal{R})})$.

Cheon et al. were able to adapt the procedure of classical homomorphic schemes to their encoding method and since under σ both additions and multiplications are coefficient-wise, homomorphic additions and multiplications act on each slot independently. In particular one is able to perform directly additions and multiplications of $n/2$ complex-values in parallel

even by using power-of-two cyclotomics. Therefore, it is more advantageous that the classical batching technique described in Section 1.3.3 which depends on the properties of the cyclotomic polynomial. However the main drawback of their construction is that it only allows to perform arithmetic on approximate numbers. From now for efficiency reasons we only consider power-of-two cyclotomics.

In the context of [CKKS17], Cheon et al. applied their encoding technique to the BGV scheme [BGV12]. The main difference between the original scheme and their adaptation is that they do not use any bound on the size of the coefficient in the plaintext space, hence in their construction the plaintext space is $\mathcal{R}$ and not $\mathcal{R}_t$. In this way the noise is not multiplied by a factor t and as a consequence, an encryption of $\boldsymbol{m} \in \mathcal{R}$ in this construction corresponds to a pair of polynomials $\mathtt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$ which, once evaluated on the secret key $\boldsymbol{s}$, satisfy:

$$[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q = \boldsymbol{m} + \boldsymbol{v} = \sigma^{-1}(\Delta \boldsymbol{z} + \boldsymbol{f}) + \boldsymbol{v} \tag{4.1.7}$$

with $\boldsymbol{v}$ the noise inherent to the ciphertext. Contrarily to [BGV12], the noise is not removed during the decryption procedure. As a consequence, this noise plus the error due to the rounding distort the result so that this construction only allows for an approximate arithmetic. However one can ensure a certain precision on the computation with the size of Δ, indeed from Equation (4.1.7) if we want to ensure p bits of precision on the result we need to choose Δ such that when decrypting:

$$\Delta \geq 2^p(\|\boldsymbol{v}\|_\infty^{can} + \|\boldsymbol{f}\|_\infty) \tag{4.1.8}$$

where $\boldsymbol{f}$ is the error coming from the rounding in Equation (4.1.6).

From there the procedures of BGV can be applied directly to this construction. The homomorphic addition of two ciphertexts corresponds to the pairwise addition of the ciphertexts' polynomials. One can also add a plaintext value directly to a ciphertext by adding it to the first coefficient of the ciphertext.

Similarly we consider two types of homomorphic multiplications: nonscalar and scalar multiplications. While the first type of multiplication involves two encrypted messages, in the second, one of the operands is known in the clear. The nonscalar multiplication of two ciphertexts $\mathtt{ct}$ and $\mathtt{ct}'$ is computed as usual: $\mathtt{ct}_{\text{mult}} \leftarrow \left(\left[\boldsymbol{c}_0 \cdot \boldsymbol{c}_0'\right]_q, \left[\boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0'\right]_q, \left[\boldsymbol{c}_1 \cdot \boldsymbol{c}_1'\right]_q\right)$. The relinearization procedure can be performed as usual, however the modulus-switching procedure is now applied both to reduce the growth rate of the norm of $\boldsymbol{v}$, due to the homomorphic multiplication, and to prevent the growth of the size of the message. Indeed, since after a multiplication, we basically obtain an encryption of $\Delta^2 \boldsymbol{z} \cdot \boldsymbol{z}'$, one should choose $\Delta \simeq q/q'$ so that after the rescaling operation, the extra Δ factor is removed.

Scalar multiplications can be computed more efficiently than nonscalar multiplications. The operand $\boldsymbol{z}'$ that is known in the clear is encoded as $\boldsymbol{m}' = \sigma^{-1}(\lfloor \Delta \boldsymbol{z}' \rceil_{\sigma(\mathcal{R})})$. Afterwards, the ciphertext $\mathtt{ct} = (\boldsymbol{c_0}, \boldsymbol{c_1})$ is multiplied by $\boldsymbol{m}'$, producing $([\boldsymbol{m}' \cdot \boldsymbol{c_0}]_q, [\boldsymbol{m}' \cdot \boldsymbol{c_1}]_q)$. Since the resulting ciphertext has only two elements, relinearisation is no longer required. However, one still needs to apply modulus-switching.

It is also useful to permute the slots of a message encrypted under a ciphertext $(\boldsymbol{c}_0, \boldsymbol{c}_1)$. In the case of power-of-two cyclotomics, there is a one-to-one correspondence between the coefficients one obtains through the canonical embedding and the coefficients of the negatively-wrapped NTT of the polynomial. Hence, changing the order of the latter, while respecting the position of the complex conjugates, will also permute the slots of the former. Therefore to permute the slots of a plaintext, one does not need to consider the action of the Galois group on the slots and can just instead compute the negatively-wrapped NTT of $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$, and changes the order of the resulting coefficients. After both $\boldsymbol{c_0}$ and $\boldsymbol{c_1}$ are permuted, the ciphertext is decipherable under the equivalently permuted $\boldsymbol{s}$ (see Section 1.3.3). Hence we need to apply a key-switching procedure as explained in Section 1.3.3 to be able to decrypt under the original secret key $\boldsymbol{s}$.

4.2 Novel homomorphic techniques for SVM classification

In the analysed scenario, two parties are considered: a client who wishes to classify sensitive data, and a service provider that offers a classification service in a confidential manner.

4.2.1 SVM confidential protocol

The service provider uses large databases to construct an SVM model. Since his objective is to provide classification as a service, the provider will be invested in preventing someone else from being able to replicate his SVM model. However, if a client that uses this service has access to a sufficient amount of samples, he will be able to replicate the SVM by using samples $\{(\boldsymbol{x}_i, y_i)\}_i$ as input to the learning algorithm. A solution to this problem might involve limiting the number of classifications a client might perform in a given period of time. This will lead to a slower leakage of data. Furthermore, the service provider should limit the number of information that is leaked with each classification. Consider, for instance, that a linear kernel is used. In this case, (4.1.1) reduces down to:

$$y = \text{sign}\left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + \beta\right) \tag{4.2.1}$$

where $\boldsymbol{w} = \sum_{i=0}^{r} \alpha_i y_i \boldsymbol{x}_i$. If simplifying assumptions were assumed, like in [GLN13], namely by not computing the sign function in (4.2.1), a client could completely determine the coefficients of $\boldsymbol{w}$ and β. He would only require $h+1$ classifications of the kind $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + \beta$ for $h+1$ linearly independent $\boldsymbol{x}$, where h denotes the number of features. The previous example shows the importance of limiting the amount of information that is provided to the client. Despite the reduced set of operations available with the considered homomorphic scheme, we will design efficient sign determination systems to adress this problem.

A second concern is that of protecting the support-vectors. Since the support-vectors encode examples from the dataset used to train the SVM, they constitute most of the time confidential data. There should therefore be mechanisms in place that prevent their disclosure. This security issue has been addressed in [BCN+14]. A randomised mechanism is used during

the learning phase that supports differential privacy - guaranteeing that even if a training datum is changed or removed, the model output will not change significantly. This concept embodies the idea that the output of the learner is an aggregate statistic of the sensitive data, and if a large amount of data is being aggregated, the statistic should not be very sensitive to changes or removal of a single datum. With this technique, [BCN+14] is able to prove security even against an attacker with knowledge of a significant set of the training data, knowledge of the learning mechanism (barring its source of randomness) and arbitrary access to the classifier, that tries to determine the features or the training label of a training datum unknown to him.

In the previous discussion, it has been assumed that the client had knowledge of the kernel function being used. While the service provider might try to hide this information from the client, so that it becomes harder for the client to mimic his model, it will not result in a considerable improvement in performance. First, there is a limited number of kernels that can be considered. Second, the service provider needs to make the scheme parameters available which, due to efficiency reasons, should be tuned for the used kernel. By taking this into consideration the client will be able to further reduce the already small number of possible used kernels.

Similarly, the client is interested in protecting his own data. Since `BGV` achieves semantic security against passive adversaries, it ensures that no adversary is capable of distinguishing an encryption of one message from another. In addition, we assume that the service provider is an honest-but-curious party, i.e. it will precisely follow the stated protocol to provide the desired functionality, but will look at the available information and try to exploit it. This assumption is a reasonable model for an economically motivated provider. The provider will be motivated to offer an excellent service, so that its reputation is not damaged, but will still take advantage of any extra information to improve its profits. A malicious provider would be a much stronger adversary, who could intentionally produce wrong results. Protecting the protocol against this type of adversary would significantly increase its complexity, making it impractical. During the protocol execution, the service provider only has access to the client's data in an encrypted form. Due to `BGV`'s semantic security, we can assume he is unable to derive any information from there.

The protocol herein proposed consists of three main phases. During the first phase, the keys underlying `BGV` are generated by the client, and the service provider is provisioned with the public-key material. Keys need only to be generated every once in a while (e.g. yearly). During the second phase, the client sends the service provider the data to be classified in encrypted format. The provider will operate on it homomorphically and return the encrypted result of the computation back to the client. In a final phase, the classification result is decrypted. In the following sections, we will not only focus on optimising the efficiency of the homomorphic evaluation of the SVM classification by the service provider, but also on limiting the amount of information that is leaked with each classification.

4.2.2 Message encoding

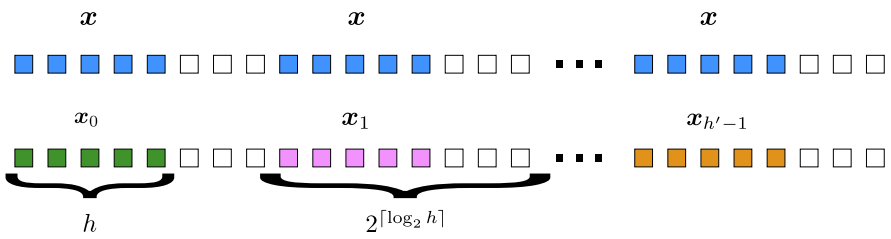


Figure 13: Mapping an input vector of features $\boldsymbol{x}$ and h' support-vectors $\boldsymbol{x_i}$ onto the plaintext space

Batching is herein exploited to optimise the efficiency of (4.1.1) by evaluating the kernel function simultaneously for several support-vectors. As previously stated, by exploiting the canonical embedding, one can encrypt $\varphi(m)/2 = n/2$ values in the same ciphertext, and apply homomorphic additions and multiplications that operate on each value independently. Since we will be working with power-of-two cyclotomics, we restrict ourselves to the case where n is a power of two. If the value slots are numbered from 0 to $n/2 - 1$, and an input vector $\boldsymbol{x}$ has $h < n/2$ features, then we map it to the plaintext space as follows:

$$\tilde{x}_{k \cdot 2^{\lceil \log_2 h \rceil} + j} = \begin{cases} x_j & \text{if } 0 \leq j < h \\ 0 & \text{otherwise} \end{cases} \tag{4.2.2}$$

The encoding in (4.2.2) replicates $n/2^{\lceil \log_2 h \rceil + 1}$ times the vector of feature $\boldsymbol{x}$, as exemplified in Figure 13 and enables batching without the client knowing the number of support-vectors that will be operated on. If the server uses more than $n/2^{\lceil \log_2 h \rceil + 1}$ support-vectors, the input ciphertext can be replicated to obtain further copies of the same value. In contrast, if fewer than $n/2^{\lceil \log_2 h \rceil + 1}$ support-vectors are employed, the unused copies of $\boldsymbol{x}$ in $\tilde{\boldsymbol{x}}$ will be multiplied by 0 during the kernel evaluation and will not influence further computations. $h' \leq n/2^{\lceil \log_2 h \rceil + 1}$ support-vectors are encoded in a single cryptogram as depicted in Figure 13 for $h' = n/2^{\lceil \log_2 h \rceil + 1}$:

$$\tilde{x}_{k \cdot 2^{\lceil \log_2 \rceil} k + j} = \begin{cases} x_{k,j} & \text{if } 0 \leq j < h, 0 \leq k < h' \\ 0 & \text{otherwise} \end{cases} \tag{4.2.3}$$

Algorithm 20 Homomorphic description of Binary Addition Tree

Require: $(\text{logSep}, \text{logAmount}) \in \mathbb{N}^2$
Require: $\text{Enc}(\tilde{\boldsymbol{x}})$
  $y \leftarrow \text{Enc}(\tilde{\boldsymbol{x}})$
  for $i \leftarrow 0$ ***to*** $\text{logAmount} - 1$ **do**
    $rotY \leftarrow \texttt{Rotate}_{2^{logSep}}(y)$
    $y \leftarrow \texttt{HomAdd}(y, rotY)$
    $\text{logSep} \leftarrow \text{logSep} + 1$
  return y

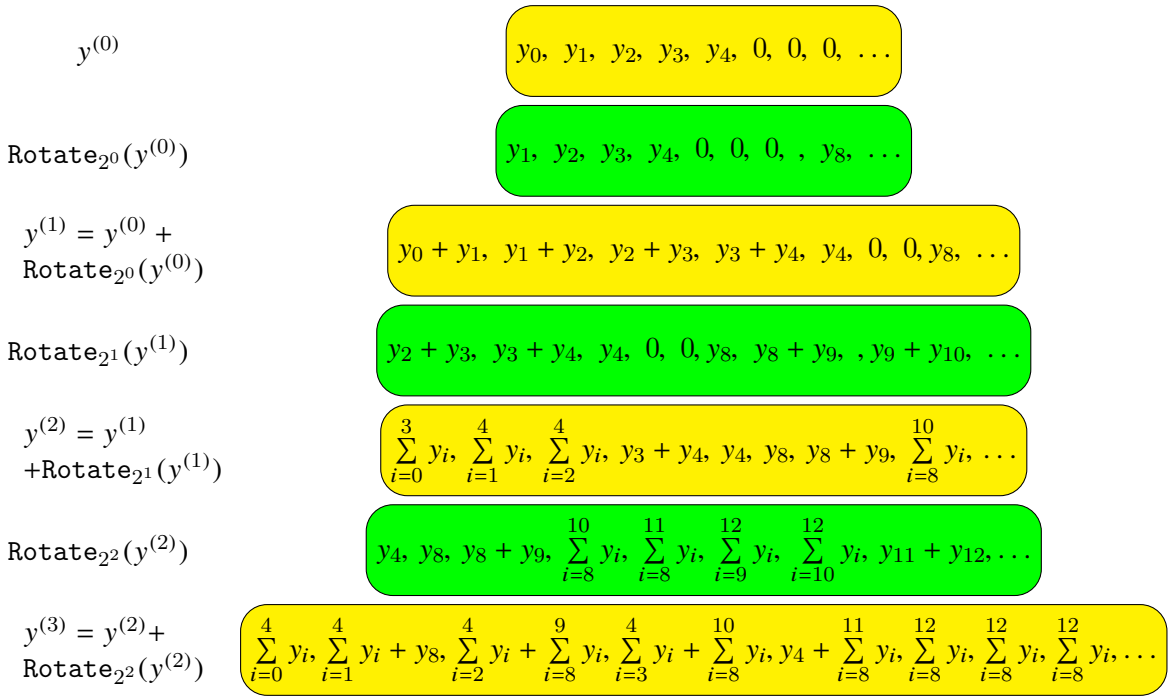


Figure 14: Illustration of the computation of a Binary Addition Tree

The computation of inner products has a central role in the evaluation of kernels. To compute the inner product of h' copies of $\boldsymbol{x}$ and h' support-vectors, one starts by homomorphically multiplying the two ciphertexts that respectively encode them. Afterwards, a binary addition tree is used to add their values, with Algorithm 20, logSep = 0 and logAmount = $\lceil \log_2 h \rceil$. In Algorithm 20, the $\texttt{Rotate}_{2^{\text{logSep}}}(y)$ function rotates the slots of y 2^{logSep} places in a circular fashion (mapping slot i to $|i - 2^{\text{logSep}}|_{n/2}$), and `HomAdd` homomorphically adds the two ciphertexts. The steps of the Algorithm are illustrated in Figure 14. We pick up the example of Figure 13, for vectors of 5 features and additions of slots for the first 5 elements are illustrated. When the function terminates, the results of the inner products will be located in the slots $k \cdot 2^{\lceil \log_2 h \rceil}$, for all $0 \leq k < h'$, of y.

After the computation of the inner product, the kernel functions operate on y without performing any rotation, producing a value z. The summation in (4.1.1) will thus require adding the slots $k \cdot 2^{\lceil \log_2 h \rceil}$, for all $0 \leq k < h'$, of z. This can be achieved with Algorithm 20 by setting logSep = $\lceil \log_2 h \rceil$ and logAmount = $\log_2 n - \lceil \log_2 h \rceil - 1$. When more than $n/2^{\lceil \log_2 h \rceil + 1}$ support-vectors are used, z will be spread over several ciphertexts. After applying Algorithm 20 to each of them, their sum is computed homomorphically.

4.2.3 Polynomial evaluation

The homomorphic evaluation of (4.1.3), (4.1.4) and (4.1.5) relies on the evaluation of polynomials. In particular, (4.1.3) can be interpreted as a polynomial with a single nonzero coefficient. Similarly, the functions $e^{-\gamma x}$ and $\tanh(x)$ in (4.1.5) and (4.1.4) are approximated by polynomials with Remez' algorithm ([Rem34]). While a truncated Taylor series would

approximate a function near a point, Remez' algorithm provides a minimax approximation of a function over an interval which is particularly useful in our setting. Hence, by establishing bounds on the magnitude of the coefficients of $\boldsymbol{x}$, we are able to bound the magnitude of the input of $e^{-\gamma x}$ and $\tanh(x)$ in (4.1.5) and (4.1.4). Therefore, we are able to maintain a small maximum error across the whole input interval. Since with a Taylor expansion the error is kept small close to the approximation point, and gets significantly larger as the points get further away from it, one would need larger degrees to keep an accurate approximation within the considered interval.

Traditional polynomial evaluation techniques, which minimise the number of nonscalar multiplications ([PS73]), have been previously considered in the context of homomorphic encryption ([CKKS17]). While this type of algorithms is optimal when computing in the clear, the same is not necessarily true when performing operations with HE. Indeed, one needs to tune the cryptographic scheme parameters to the multiplicative depth of the circuit one wants to evaluate. Having larger parameters leads to a severe degradation of performance. Hence, one can no longer consider the number of multiplications as a whole, but one has rather to consider the maximum number of multiplications in a computational path.

Algorithm 21 Homomorphic evaluation of a polynomial

Require: chains $\leftarrow \{2 : (1,1), 3 : (1,2), \ldots\}$
Require: Enc($\boldsymbol{x}$)
Require: $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_d$
  $y \leftarrow \texttt{HomAdd}((\boldsymbol{p}_0, 0), \texttt{HomMul}(\text{Enc}(\boldsymbol{x}), (\boldsymbol{p}_1, 0)))$
  powersOfX $\leftarrow \{1 : \text{Enc}(\boldsymbol{x})\}$
  for $i \leftarrow 2$ ***to*** d **do**
    $(a_j^{(i)}, a_k^{(i)}) \leftarrow$ chains$[i]$
    $x^{a_j^{(i)}} \leftarrow$ powersOfX $\left[a_j^{(i)}\right]$
    $x^{a_k^{(i)}} \leftarrow$ powersOfX $\left[a_k^{(i)}\right]$
    $x^i \leftarrow \texttt{HomMul}\left(x^{a_j^{(i)}}, x^{a_k^{(i)}}\right)$
    $y \leftarrow \texttt{HomAdd}\left(y, \texttt{HomMul}\left(x^i, \boldsymbol{p}_i\right)\right)$
    if $a_j^{(i)}$ does not belong to any pair in $chains[i+1, \ldots, d]$ **then**
      powersOfX $\leftarrow$ powersOfX$\setminus \left\{a_j^{(i)} : x^{a_j^{(i)}}\right\}$
    if $a_k^{(i)}$ does not belong to any pair in $chains[i+1, \ldots, d]$ **then**
      powersOfX $\leftarrow$ powersOfX$\setminus \left\{a_k^{(i)} : x^{a_j^{(i)}}\right\}$
    if i belongs to a pair in $chains[i+1, \ldots, d]$ **then**
      powersOfX $\leftarrow$ powersOfX $\cup \left\{i : x^i\right\}$
  return y

Algorithm 21, herein proposed for the evaluation of a polynomial of degree d, minimises the circuit multiplicative depth by precomputing the minimum addition chain for each $2 \leq n \leq d$. In the Algorithm, the notation $d = \{k_1 : i_1, k_2 : i_2, \ldots\}$ is used to denote a dictionary with elements $i_1, i_2, \ldots$, that can be accessed by the keys $k_1, k_2, \ldots$, via the $[\cdot]$ operator: $d[k_1] =$

$i_1, d[k_2] = i_2, \ldots$ Moreover, the notation $d[k_1, k_2, \ldots]$ is used to denote the multiset $\{i_1, i_2, \ldots\}$. An addition chain corresponds to a sequence of integers $(a_0, \ldots, a_s)$ such that $\exists_{0 \leq j,k<i} a_i = a_j + a_k$ for $i > 0$ and $a_0 = 1$. A minimum addition chain of n corresponds to a shortest sequence such that $a_s = n$. By using a minimum addition chain $(a_0, \ldots, a_s)$ of n, one is capable of evaluating x^n with the minimum multiplicative depth, by computing:

$$x^{a_i} = x^{a_j} \times x^{a_k}, 0 < i \leq s, a_i = a_j + a_k \tag{4.2.4}$$

The proposed procedure computes x^n starting at $n = 2$ and ending at $n = d$. Each time a power n of x is computed, x^n is stored if n is required by an addition chain of $n' > n$. Furthermore, one only needs to store the final pair $a_j + a_k = n$ for each $2 \leq n \leq d$, since the values of x^{a_j} and x^{a_k} will have been computed previously in the algorithm.

Finally, each time a power of x is computed its value is scaled and added to an intermediate sum:

$$p(x) = \sum_{i=0}^{d} p_i x^i \tag{4.2.5}$$

4.2.4 Sign evaluation

The Newton-Raphson method is a technique to find successively better approximations to roots of real-valued functions. Herein, this method is applied to homomorphically compute the sign of a number. The proposed procedure approximates the roots of $f(y) = \frac{1}{y^2} - 1$, converging to -1 for an initial negative guess and to $+1$ for an initial positive value. For an input value $y_0 = y$ a succession $y_1, \ldots, y_t$ is computed such that:

$$y_{i+1} = y_i - \frac{f(y_i)}{f'(y_i)} = y_i \left(\frac{3 - y_i^2}{2} \right) = g(y_i) \tag{4.2.6}$$

Lemma 4.2.1. The recurrent sequence defined by (4.2.6) converges to -1 (resp. 1) if $y_0 \in (-\sqrt{3}, 0)$ (resp. $y_0 \in (0, \sqrt{3})$) and is constant equal to 0 from rank $n = 1$ if $y_0 \in \{-\sqrt{3}, 0, \sqrt{3}\}$. Moreover the convergence to ± 1 is quadratic.

Proof. Figure 15 presents the variations of the function g, defined by (4.2.6), on $[-\sqrt{3}, \sqrt{3}]$.

y	$-\sqrt{3}$	-1	0	1	$\sqrt{3}$
var. of g	0 ↘	-1 ↗	0 ↗	1 ↘	0

Figure 15: Variations of the function g

One can see that $g([-1, 0]) = [-1, 0]$ (resp. $g([0, 1]) = [0, 1]$) and that $g([-\sqrt{3}, 1]) = [-1, 0]$ (resp. $g([1, \sqrt{3}]) = [0, 1]$). Therefore if $y_0 \in [-\sqrt{3}, \sqrt{3}]$, all the y_is, for $i \geq 1$, will belong to $[-1, 1]$. The function g is increasing on $[-1, 1]$, thus the sequence defined by $y_{n+1} = g(y_n)$ and

$y_0 \in [-\sqrt{3}, \sqrt{3}]$ is monotonic from rank $n = 1$ and since it is bounded and g is continuous, it converges to a limit $\ell \in \{-1, 0, 1\}$ which is a fixed point of g.

Studying the sign of $g(y) - y$ shows that the sequence $(y_n)_{n\in\mathbb{N}}$ is decreasing (resp. increasing) from $n = 1$ if $y_0 \in (-\sqrt{3}, 0)$ (resp. $y_0 \in (0, \sqrt{3})$) and thus converges to -1 (resp. 1). The case where $y_0 = 0$ is trivial since $g(0) = 0$ and if $y_0 = \pm\sqrt{3}$, then $y_1 = 0$.

Even though it is well known that the Newton-Raphson method converges quadratically, in our case it can be deduced directly from:

$$\frac{|g(y_n) - \ell|}{|y_n - \ell|^2} = |y_n/2 + \ell| \underset{n\to+\infty}{\longrightarrow} 3/2 \text{ for } y_n \underset{n\to+\infty}{\longrightarrow} \ell \in \{-1, 1\}.$$

□

Following Lemma 4.2.1, if we assume a certain bound on the coefficients of $\boldsymbol{x}$ in (4.1.1), one can compute a scaling factor $s > 0$ such that:

$$-\sqrt{3} < s\left(\sum_{i=0}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + \beta\right) < \sqrt{3} \tag{4.2.7}$$

to ensure the convergence of the sign function in (4.1.1) without changing the final result.

4.2.5 Leakage of information

Ideally, the server that homomorphically evaluates the SVM would return nothing but the encrypted sign of y in order to avoid leaking any information about its model. The encoding procedure of Cheon et al. [CKKS17] allows to encrypt up to $n = \varphi(m)$ complex values per plaintext, which permits, as described in Section 4.2, to perform approximated computations on complex values efficiently. However since the sign of these values is not encoded in a specific bit or slot it cannot be extracted directly. As a consequence, we choose to use the iterative Newton-Raphson method described in section 4.2.4, so that after t iterations one will obtain $y_t = g^t(y_0)$ which will be close to ± 1 depending on the sign of y. However, since y_t will not be exactly ± 1 we analyse in this section how much information about y_0 is leaked with this approach.

Direct approach

One can see from Figure 15 that g maps $[-1, 1]$ to $[-1, 1]$ with a one-to-one correspondence. Hence, assuming that the result y is computed with enough precision, a malicious client could invert g on $[-1, 1]$, and thus retrieve y_1 by applying the inverse of g $t - 1$ times. Since y_0 can belong to either $[0, 1)$ (resp. $(-1, 0]$) or $[1, \sqrt{3})$ (resp. $(-\sqrt{3}, -1]$), the client will only have to choose between two possible values for y_0. Moreover, since the initial values y_0 closer to 0 or $\pm\sqrt{3}$ converge more slowly to ± 1, these values are easier to distinguish from the others.

A straightforward way to prevent the inversion of the method would be to perform the computations with a lower precision, by choosing a smaller scaling factor Δ for the encoding (cf. Equation 4.1.8). Although this precision is in practice the key point to know whether

or not a malicious client can invert the process to retrieve the original value y_0, reducing it would result in a service of lower quality. Indeed, the wrong sign value would possibly be computed for $|y| < 2^{-p}$, and as a consequence the corresponding input $\boldsymbol{x}$ would be interpreted as belonging to the wrong class. Therefore, this approach should not be considered as a viable solution.

Since the convergence to ± 1 is quadratic, we can assume that after t iterations the values close enough to ± 1 loose t^2 bits of precision. Thus if the values are encoded with 53 bits of precision, which corresponds to double precision for a floating point representation, running $t = 7$ or 8 iterations would be enough to prevent a malicious client to invert the process for values of y_0 not too close to 0 or $\pm\sqrt{3}$.

Randomised approach

One way to prevent a malicious client from retrieving the initial values would be to return a "random" value of the same sign as the real output. However, this is hardly achievable since the server does not know the sign of y. Nevertheless, it is possible to slightly randomise the previous approach so that it will be harder for a malicious client to invert the process. Indeed, one can choose to add a random noise ε_i uniform in $[0, 2^{-i-1}]$ at the i^{th} iteration of the previous method so that:

$$y_{i+1} = g(y_i) + \varepsilon_{i+1} g(y_i) = \tilde{g}(y_i) \text{ with } \varepsilon_{i+1} \xleftarrow{\$} \left[0, 2^{-i-1}\right] \tag{4.2.8}$$

since, for any $y \in [-\sqrt{3}, \sqrt{3}]$, $g(y) \in [-1, 1]$ (cf. Figure 15), we can add to $g(y)$ a value of the same sign and with an absolute value smaller than $\sqrt{3} - 1$ without changing the sign of the output. However, in order to avoid slowing down the convergence of values close to ± 1, we need to be sure that $y_{i+1} = \tilde{g}(y_i)$ remains far enough from $\pm\sqrt{3}$, so that $\tilde{g}(y_{i+1})$ remains far enough from 0. Hence, we have chosen the ε_is smaller than $1/2^i$, which seems to be a good trade-off experimentally. Note that, because $\varepsilon_i \to 0$, the method still converges to ± 1.

The benefits of this approach are twofold: first, the output is now randomized and the extra-computations to perform this randomisation bring extra errors resulting in additional loss of precision; second, the sequence moves away from 0 faster and thus the range of values getting close to ± 1 after t iterations slightly increases. Moreover, the values close to 0 or $\pm\sqrt{3}$ are more sensitive to the perturbations brought by the ε_is, which makes them harder to distinguish from the other values and to be retrieved from y_t.

Even if it was assumed that computations were performed with an infinite precision, i.e. the ideal case for a malicious client, it would be harder to retrieve the initial values with randomisation than with the regular method. While a malicious client could perform multiple requests for the same $\boldsymbol{x}$ in order to deduce the average value $\overline{y_t}$ whose distribution he knows. In this case, since the values can be greater than 1 at each iteration step, he would have to determine at each inversion step whether the previous value was smaller or larger than 1 in absolute value, because there is no longer a one-to-one correspondence. As a consequence, by randomising the output one can use a smaller amount of iterations than for the regular case.

Practical Considerations

In practice, performing one iteration of the previous methods consumes 2 levels of multiplicative depth. Even though choosing parameters for a large multiplicative depth results in a significant loss of performance, iterating a larger number of times leads to a smaller leakage of information. The service provider has therefore to deal with a trade-off between the efficiency of his service and how much data is leaked. We have seen that for a precision of 53 bits, 7 or 8 iterations of the non-randomised sign determination method, and so also of the randomised one, should ensure the privacy of almost all the ys. However, in practice one can choose a smaller number of iterations, especially if using the randomised method, without significantly affecting the leakage of information.

Figures 16 illustrates the convergence of the two methods for different numbers of iterations, and also presents the standard deviation of the values computed with the randomised method. Herein, we have chosen to use 4 iterations of the randomised method for the SVM classification requiring therefore 8 additional levels of multiplicative depth.

4.2.6 Overall system

When considering a linear kernel in the context of (4.1.1), classification boils down to:

$$y = \text{sign}\left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + \beta\right) \tag{4.2.9}$$

where $\boldsymbol{w} = \sum_{i=0}^{n} \alpha_i y_i \boldsymbol{x}_i$. In this case, if the number of features h is smaller than $\varphi(m)/2$, $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + \beta$ can be computed with a single homomorphic multiplication, followed by $\lceil \log_2 h \rceil$ permutations and additions. Afterwards, the sign of the result is approximated.

For the polynomial and sigmoid functions, the $\gamma \boldsymbol{x}_i$ are precomputed, and the values of $\langle \gamma \boldsymbol{x}_i, \boldsymbol{x} \rangle$ are computed for $\varphi(m)/2^{\lceil \log_2 h \rceil + 1}$ support-vectors at a time using batching with the approach described in Section 4.2.2. Then, ρ is added to the results. A similar reasoning can be followed when applying the RBF kernel, namely by adding the results of the coefficient-wise product $\left(x_j - x_{i,j}\right)\left(x_j - x_{i,j}\right)$ with Algorithm 20. Afterwards, the function x^d, $\tanh(x)$ or $e^{-\gamma x}$ is applied for the polynomial, sigmoid or RBF kernel, respectively, with the techniques proposed in Section 4.2.3. The constants $\alpha_i y_i$ are multiplied by the result of $K(\boldsymbol{x}, \boldsymbol{x}_i)$, and the results are added with the binary addition tree proposed in Section 4.2.2. Finally, the bias term β is added to the intermediate result, and the sign approximation function is applied.

Even though batching is used, the server will only be interested in returning the result of one slot to the client, since, if no care is taken, the remaining slots might contain sensitive information. Hence when applying the sign evaluation technique, multiplicative constants should be encoded in a single slot (e.g. $(3, 0, \ldots, 0)$), so that information in the other slots is removed. This prevents further information from being disclosed other than the y class.

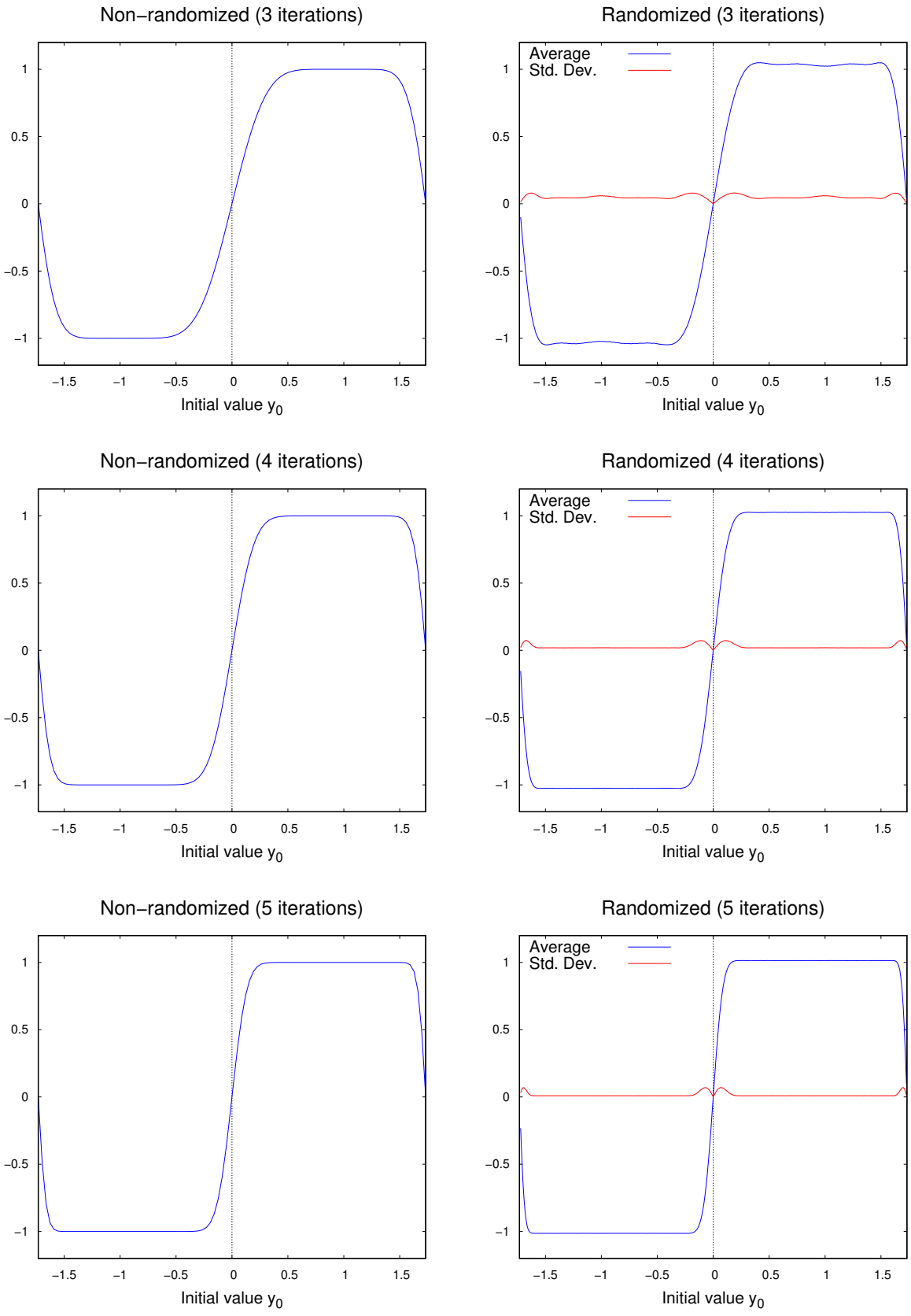


Figure 16: Output of regular and randomised methods run on plaintext values for 3, 4 and 5 iterations. Values of the randomised method correspond to an average over 2^{12} tests.

4.3 Implementation details

In this section, we propose algorithms for implementing the presented techniques on parallel systems. In particular, we will assume the nomenclature and the parallel computing model of OpenCL.

4.3.1 OpenCL parallel computing model

The OpenCL computing model considers a host connected to one or multiple *compute devices* [SGS10]. These compute devices may be Central Processing Units (CPUs), Graphics Processor Units (GPUs), among others. The compute device is composed of multiple *compute units* - which may be Streaming Processors (SPs) in NVIDIA GPUs, or cores in CPUs - which are further divided into *processing elements* - corresponding for instance to CUDA cores in NVIDIA GPUs or Single Instruction Multiple Data (SIMD) channels in CPUs.

A compute device is programmed using a C-like language. Functions signaled as kernels correspond to entry-points to the code execution. A kernel is replicated thousands of times and executed in parallel in the compute device. Each of these instances, designated a work-item, has a unique identifier, enabling them to access and process different data from one another. Moreover, and up to the extent allowed by the compute device, work-items may be grouped into work-groups. Whereas it is possible to synchronise work-items in the same work-group, the same thing is not possible for work-items across different work-groups. Similarly, whereas global memory can be accessed by all work-items, a faster type of memory, designated local memory, can be used to share data at the work-group level.

4.3.2 Architecture design

Cheon et al. have provided a generic implementation of their scheme [Che16] based on the `NTL` library. The ciphertext modulus q was chosen as a power-of-two for cheap modular reductions. However, this removes the possibility of using the efficient negatively-wrapped NTT, resulting in poor performance. In this work we have chosen to use the RNS representation of coefficients in order to make use of parallel architectures, such as GPUs.

We have chosen q as a product of 62-bit prime numbers in a similar fashion to [AMBG+16]. This allows us to represent a polynomial $a \in \mathcal{R}_q$ for a $q = q_1 \times \ldots \times q_k$ as a set of polynomials $a_i \in \mathcal{R}_{q_i}$ for all $1 \leq i \leq k$. Furthermore, we apply the negatively-wrapped NTT to polynomials. Due to this representation, polynomial operations are independents in nature and therefore readily parallelizable. More concretely, they leverage two levels of parallelism: the first derives from the independence of the polynomial coefficients, while the second arises from the independence of the RNS channels.

Additionally, negatively-wrapped NTTs need to be applied during relinearisation, modulus-switching and decryption. Since the NTT is the most burdensome operations of homomorphic computations, it is one of main targets for optimisation. Figure 17 shows how its computation is structured. For illustrative purposes, a radix-2 NTT of eight points is considered, when

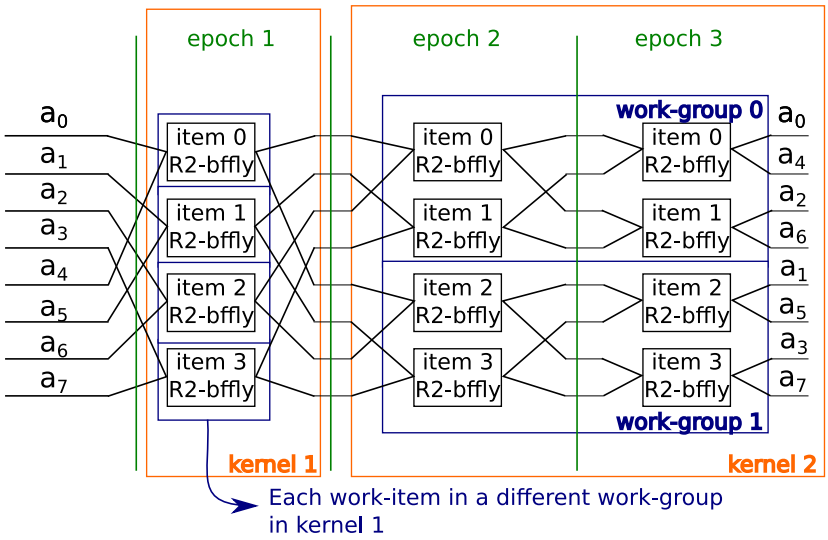


Figure 17: NTT Work Distribution

executing on a fictional compute device that allows for work-groups with a maximum of two work-items. The NTT is computed in 3 epochs, where each one of 4 work-items computes a radix-2 butterfly at each epoch. Each butterfly is comprised of a 2-point Discrete Fourier Transform (DFT), followed by a multiplication of one of the outputs by a precomputed power of an 8^{th} primitive root of unity.

One can see in Figure 17 that in the second epoch, the computation of work-item 0 depends on the output of work-item 2 in the first epoch, and that in the third epoch the computation of work-item 1 depends on the output of work-item 0 in the second epoch. Since it is not possible to synchronize 3 work-items in the considered device (since it has a maximum work-group size of two work-items), one has to execute the first epoch in a different kernel (kernel 1) from the two last epochs (kernel 2). These two kernels are queued sequentially, and whereas for kernel 1 one sets a work-group size of 1 work-item, for kernel 2 one defines a work-group size of 2 work-items. Furthermore, in the latter kernel, a synchronization point is added to each epoch, and local memory is used for faster accesses.

The example above is generalizable to r-radix NTTs of n points, wherein the initial epochs are executed on independent kernels, until the number of dependencies is confined to a number of work-items smaller or equal to the maximal work-group size supported by the compute device, and from then on computation is limited to the compute device. This technique minimizes the number of synchronizations between the host and the compute device, and generally contributes significantly to the total execution time.

4.3.3 Global strategy

We make use of the relinearization and key-switching procedures proposed in Chapter 2 which are optimised for RNS representations. During the modulus-switching procedure, the ciphertexts are scaled downed to a smaller ring. For a ciphertext whose elements are contained in $\mathcal{R}_q$ for $q = q_1 \times \ldots \times q_k$, we scale them down to the ring $\mathcal{R}_{q'}$ with $q' = q_1 \times \ldots \times q_{k-1}$. This

procedure can be efficiently computed in the RNS domain. Since the value $[a]_{q_k}$ is readily available, the scaling of $a \in \mathcal{R}_q$ can be computed as $\left[\frac{a-[a]_{q_k}}{q_k}\right]_{q'}$. As stated in Section 4.1.3, one should choose Δ as $\Delta \simeq q/q'$. In order to achieve this, we have chosen Δ as q_1 since $q_1 \simeq \ldots \simeq q_k$.

Relinearisation accounts for most of the complexity because it requires k^2 NTT (cf. Section 2.3.4). First, an inverse-NTT is applied to $\boldsymbol{a}$ so that $\mathcal{D}_{\mathtt{RNS},q}(\boldsymbol{a})$ can be computed. Then, a forward-NTT has to be applied to each $\left(\left|\boldsymbol{a}\frac{q_1}{q}\right|_{q_1}, \ldots, \left|\boldsymbol{a}\frac{q_k}{q}\right|_{q_k}\right)$, so that they can be multiplied by the relinearisation-key. We have reduced the number of times the relinearisation operation was executed, by only applying it at the beginning of the next homomorphic multiplication operation, and by performing homomorphic additions on "non-linear" elements by adding the three corresponding polynomials. In particular, for the homomorphic evaluation of an expression such as $x_0x_1 + x_2x_3$ and denoting by $(\boldsymbol{c}_{(x_i,0)}, \boldsymbol{c}_{(x_i,1)})\ \forall i \in \{0,1,2,3\}$ the "linear" encryption of x_i, one would first compute

$$\boldsymbol{c}_{x_ix_{i+1}} = (\boldsymbol{c}_{(x_i,0)}\boldsymbol{c}_{(x_{i+1},0)}, \boldsymbol{c}_{(x_i,0)}\boldsymbol{c}_{(x_{i+1},1)} + \boldsymbol{c}_{(x_{i+1},0)}\boldsymbol{c}_{(x_i,1)}, \boldsymbol{c}_{(x_i,1)}\boldsymbol{c}_{(x_{i+1},1)})\ \ \forall i \in \{0,2\}\ .$$

Afterwards, one would add the two encryptions, producing:

$$\boldsymbol{c}_{x_0x_1+x_2x_3} = (\boldsymbol{c}_{(x_0x_1,0)} + \boldsymbol{c}_{(x_2x_3,0)}, \boldsymbol{c}_{(x_0x_1,1)} + \boldsymbol{c}_{(x_2x_3,1)}, \boldsymbol{c}_{(x_0x_1,2)} + \boldsymbol{c}_{(x_2x_3,2)}).$$

If this result were applied to another homomorphic multiplication, it would require applying a relinearisation operation to bring the number of elements of $\boldsymbol{c}_{x_0x_1+x_2x_3}$ back to two. Nevertheless, one relinearisation operation was saved in comparison to a straightforward implementation.

Another important aspect with regards to delaying relinearisation is how this strategy interplays with the permutation of the batching slots. During the permutation procedure, key-switching needs to be applied, introducing noise that has the same magnitude as Δ. If this procedure were applied directly to a permuted "linear" ciphertext, the introduced noise could be large enough to change the most-significant bits of the encrypted message. We mitigate this problem with two approaches. When permuting a three-element ciphertext, we first relinearise it but do not apply modulus-switching. In this case, the message has a multiplicative factor of Δ^2, and the introduced noise will be small when compared with Δ^2. When permuting a two-element ciphertext, we first multiply it by it by an encoding of '1', implicitly changing the multiplicative factor to Δ^2, so that the previous reasoning about reducing the impact of the introduced noise is valid. In order to remove the extra Δ factor, one needs to apply modulus-switching after the key-switching procedure. We do so lazily, following a similar reasoning to the previous one that allowed us to reduce the number of relinearisations: when adding several permuted ciphertexts, we only apply modulus-switching at the end of the sum.

4.4 Experimental results

In this section, we experimentally evaluate the proposed methods, and compare them with state of the art. The characteristics of the considered platforms can be found in Table 12. We mostly use the GTX 980 as an archetype of the kind of parallel platforms that can be found in High Performance Computing (HPC) servers to compare different methods; use the GTX 680 to analyse the scalability of the proposed techniques across different NVIDIA micro-architectures; and use the i7-5960X for a sequential reference implementation.

Platform	Micro-architecture	Nbr. of threads/ CUDA cores	Frequency (MHz)	Main Memory (GB)
NVIDIA GTX 980	Maxwell	2048	1126	4
NVIDIA GTX 680	Kepler	1536	1006	2
Intel i7-5960X	Haswell	16	3000	32

Table 12: Experimental Setup

4.4.1 Polynomial evaluation

The method proposed in [PS73] achieves the minimum number of nonscalar multiplications for polynomial evaluation. Since this type of multiplication is considerably more expensive than scalar multiplications, [PS73] is a good candidate for the homomorphic evaluation of polynomials. Both the method described in Section 4.2.3 and the one described in [PS73] were implemented in an NVIDIA GTX 980 GPU exploiting HE. When implementing [PS73] to evaluate a polynomial of degree d, m_1 was chosen to be the largest integer dividing $d+1$ and smaller than $\sqrt{d+1}$. Furthermore, m_2 was defined as $m_2 = \frac{d+1}{m_1}$. Initially, the first m_1 powers of the input x are computed by repeated multiplication. Afterwards, the $d+1$ coefficients of the polynomial are split into m_2 polynomials each of degree $m_1 - 1$:

$$p(x) = \sum_{i=0}^{d} p_i x^i = \sum_{i=0}^{m_2-1} \left(\sum_{j=0}^{m_1-1} p_{im_1+j} x^j \right) (x^{m_1})^i = \sum_{i=0}^{m_2-1} g_i(x) z^i \tag{4.4.1}$$

where $g_i(x) = \sum_{j=0}^{m_1-1} p_{im_1+j} x^j$ and $z = x^{m_1}$. The polynomials g_i were evaluated with $m_1 - 1$ scalar multiplications, since the values of $x, \ldots, x^{m_1-1}$ had been previously computed. The computation of p was finalised using Horner's method with $m_2 - 1$ non-scalar multiplications.

The efficiency of the two evaluation approaches was tested for polynomials of degree $d \in \{4, 8, 16, 32, 64\}$ with the coefficients uniformly sampled from the $[-1, 1]$ interval, and the results were reported in Table 13. As the multiplicative depths of the methods gets larger, larger values of q need to be used, so that the scheme can cope with the extra noise generated by the homomorphic multiplications. This leads to a need of increasing the value of m that defines the underlying cyclotomic ring, so as to ensure security. The performance of the polynomial evaluation methods is severely degraded as these two parameters increase. Even though [PS73] aims at minimising the number of non-scalar multiplications, by using

Method	Degree	Mult. Depth	m	$\log_2 q$	# scalar multiplications	# non-scalar multiplications	Execution Time (s)
Proposed	4	3	2^{14}	310	3	2	0.045
[PS73]	4	4	2^{15}	372	0	3	0.070
Proposed	8	5	2^{15}	434	7	6	0.34
[PS73]	8	4	2^{15}	372	6	3	0.15
Proposed	16	6	2^{15}	496	15	14	0.82
[PS73]	16	16	2^{16}	1116	0	15	2.23
Proposed	32	8	2^{15}	620	31	30	2.24
[PS73]	32	12	2^{16}	868	22	11	3.28
Proposed	64	9	2^{15}	682	63	62	4.72
[PS73]	64	16	2^{16}	1116	52	15	12.23

Table 13: Efficiency of Homomorphic Polynomial Evaluation

more efficient scalar multiplications, its multiplicative depth increases at a faster pace than the proposed method, leading to the poor scalability of [PS73], as it is clearly observable in Table 13. In addition, [PS73] performs worst when $d+1$ is prime, since this will lead to $m_1 = 1$ and $m_2 = d + 1$, contributing to larger multiplicative depths.

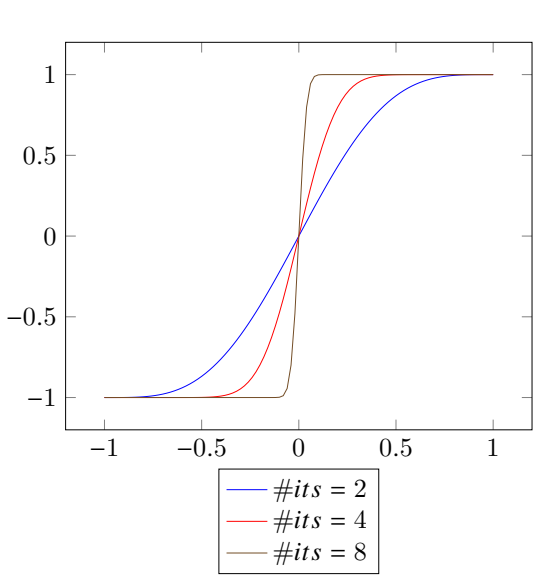
4.4.2 Sign evaluation

The efficiency of the proposed sign evaluation method, when compared to that of [CDSM15] was similarly assessed in an NVIDIA GTX 980, and the results were reported in Table 14. When using the method proposed in [CDSM15], the sign of an input x is approximated by a sum of sine functions which are approximated by polynomials:

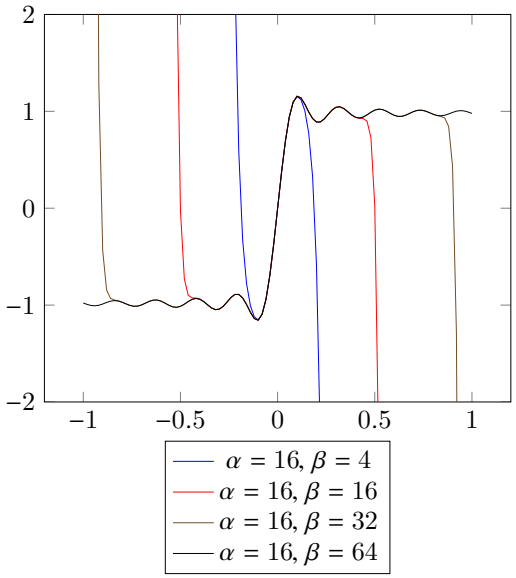
$$\text{sign}(x) \approx \frac{1}{0.8} \sum_{j=1}^{\beta} \sum_{i=1}^{\alpha} \frac{(-1)^{j-1}(2i-1)^{2j-2}}{(2j-1)!} x^{2j-1} \tag{4.4.2}$$

The polynomial described in (4.4.2) was evaluated with the method proposed in Section 4.2.3. Random values in the $[-1, 1]$ interval were generated and the mean and the standard deviation of the absolute value of the results were computed for several iterations of the two proposed methods, and for different values of β and $\alpha = 16$ for the method in [CDSM15] (α does not influence the execution time and, as long as it is sufficiently large, it does not significantly change the accuracy of the result). The parameters m and q were selected to be in the same order of magnitude for both approaches. Ideally, one should achieve a mean absolute value of '1' and a small standard deviation. Since the approximation in (4.4.2) converges slowly, one needs large values of β to achieve significant numeric results. The difference in convergence rates is highlighted in Figures 18a and 18b. Since the proposed methods offer a better approximation in the whole domain, while [CDSM15] focus on optimising the approximation around the origin, one needs a larger degree for the latter method to cover a more significant interval which negatively affects its performance. Finally, one can see that the randomisation mechanisms, used to reduce information leakage, impose a small overhead, on average 11%; while at the same time improving both the mean and the standard deviation

due to a faster convergence of the values close to 0.



(a) Proposed Non-Randomised Sign Evaluation Method



(b) Sign Evaluation Method Proposed in [CDSM15]

Figure 18: Comparison of the convergence rates of (a) the sign approximation method proposed herein and (b) that of [CDSM15]

Method	# of its/ β	Mult. Depth	m	$\log_2 q$	Mean of Abs. Value	Std. Dev. of Abs. Value	Execution Time (s)
Regular	2	4	2^{15}	372	0.73	0.30	0.076
Regular	4	8	2^{15}	620	0.87	0.24	0.32
Regular	8	16	2^{16}	1116	0.98	0.12	3.31
Noisy	2	4	2^{15}	372	0.85	0.32	0.089
Noisy	4	8	2^{15}	620	0.94	0.21	0.35
Noisy	8	16	2^{16}	1116	0.99	0.10	3.54
[CDSM15]	4	5	2^{15}	434	69957	128272	0.29
[CDSM15]	16	8	2^{15}	620	5.64×10^{7}	1.55×10^{8}	0.75
[CDSM15]	32	9	2^{15}	682	543	3066	4.64
[CDSM15]	64	11	2^{16}	806	0.97	0.45	23.33

Table 14: Efficiency of Homomorphic Sign Evaluation

4.4.3 Global procedure

The LIBSVM library [CL11] was used to train models for the Statlog Australian Credit Card Approval dataset [Fan], the Connectionist Bench (Sonar, Mines vs. Rocks) dataset [Fan], the Pima Indian Diabetes dataset [Fan] and the Statlog Heart Data Set [Fan] with the characteristic described in Table 15. Whereas with the previous experimental results we were concerned with comparing the performance of the proposed methods with those of related art, in Table 17 we present, for the first time, the performance of homomorphically performing an SVM classification. We have chosen a standard deviation $\sigma_{\text{err}} = 6.0$ for the error distribution, the

secret key has been chosen with 64 non-zero coefficients sampled uniformly in $\{\pm 1\}$, and the level of security λ achieved by our parameters was computed using the online estimator provided by [APS15]. We also present the size of the keys (relinearization and key-switching) the client has to upload to the server before starting the protocol together with the size of the ciphertexts. Remember that we are in an amortised model, so the client has only to provision the keys once for a long period during which he can perform several requests.

Dataset	# of features	Training Dataset Size	Testing Dataset Size	Kernel
Australian	14	400	190	Linear
Sonar	60	160	48	Polynomial (degree 5)
Diabetes	8	600	168	Sigmoid
Heart	13	200	70	RBF

Table 15: Datasets to Test Private SVM Classification

We assess the efficiency of the OpenCL implementation on two generations of NVIDIA GPUs, in order to evaluate its scalability, and compare it with a reference C++ sequential implementation on an Intel i7-5960X. The sign function in (4.1.1) was evaluated by iterating the method proposed in Section 4.2.5 4 times. The sigmoid and exponential functions in (4.1.4) and (4.1.5) were approximated by polynomials presented in Table 16.

index	Coefficients	
	$x \mapsto \tanh(x)$	$x \mapsto e^{-\gamma x}$
	$a = 0.51225306238208002$	$a = 115$
0	1.4009642707414478e-17	1.0000000000000002
1	0.99999999999995082	-0.00048828100000000192
2	-2.1227219660455179e-14	1.1920916748065294e-07
3	-0.33333333332155596	-1.9402523836879882e-11
4	1.9300123814730566e-12	2.3684705729457643e-15
5	0.1333333325110871	-2.3129124018455077e-19
6	-6.5224256206901339e-11	1.9064448146349986e-23
7	-0.053968227901277463	-4.4241699170007713e-27
8	1.1171625055134562e-09	-7.4394009787886375e-29
9	0.021869040479270206	9.4733594535281552e-31
10	-1.0923689311744991e-08	1.2314188912818555e-32
11	-0.0088586960556909129	-1.5386535912138098e-34
12	6.3527986719020514e-08	-1.1349649205298355e-36
13	0.0035641099208678808	1.3779247892196034e-38
14	-2.170761532109173e-07	5.6034063188191365e-41
15	-0.0013512032562647397	-6.4143854972872915e-43
16	4.0192953621050575e-07	-1.2528959902659904e-45
17	0.00036617202103725164	1.2116502342840465e-47
18	-3.108838239985658e-07	6.3734555982768629e-51

Table 16: Polynomials of degree 18, computed with Remez algorithm, used for approximating tanh and $e^{-\gamma}$ on the interval $[-a, a]$ with $\gamma = 0.00048828100000000002$.

The accuracy achieved during the homomorphic test was exactly the same as that achieved

with unencrypted values, leading to the conclusion that the errors introduced by the approximate arithmetic of [CKKS17] and our polynomial approximations are not significant. Finally, an average speedup of 2.70 is obtained when comparing the parallel GTX 980 implementation with the sequential i7-5960X. This is made possible by the exploitation of the RNS, which makes the computation of the several work-items uniform and hence more suitable to implementation on GPUs. The improvement in performance that is obtained when comparing the execution on the GTX 980 to that of GTX 680 shows that the developed OpenCL system is very scalable with respect to the number of available processing elements, since the GTX 980 has 2048 CUDA cores, while the GTX 680 has 1536 CUDA cores; and also that it is capable of taking advantage of the reduced latency of the integer arithmetic of the Maxwell micro-architecture in comparison to the Kepler micro-architecture.

dataset	kernel	mult. depth	m	$\log_2 q$	λ (bits)	`keys` (GB) / `ct` (MB)	accuracy (%)	platform	exec. time (s)
Australian	(4.1.2)	9	2^{15}	682	80	0.43 / 3	86.32	GTX 980 GTX 680 i7-5960X	1.14 2.80 3.32
Sonar	(4.1.3)	13	2^{16}	930	100	1.71 / 8	89.58	GTX 980 GTX 680 i7-5960X	13.28 8.68 25.73
Diabetes	(4.1.4)	16	2^{16}	1116	89	2.47 / 9	76.79	GTX 980 GTX 680 i7-5960X	21.14 51.17 64.33
Heart	(4.1.5)	16	2^{16}	1116	89	2.47 / 9	81.43	GTX 980 GTX 680 i7-5960X	21.42 52.11 66.08

Table 17: Performance of the Proposed Homomorphic SVM Classification

In [GLN13], the Linear Means (LM) and Fisher's Linear Discriminant (FLD) classifiers were considered. Both methods compute the most likely class y based on the input vector of features $\boldsymbol{x}$ as follows:

$$y = \text{sign}\left(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + \beta\right) \tag{4.4.3}$$

where $\boldsymbol{w}$ is learned during the training phase. However, in [GLN13], the computation of the sign function is deferred until after decryption. Doing so would not be acceptable in our setting, because it might lead to a larger information leak. Homomorphic classifications take up to 6 seconds with the LM classifier on data with 30 features, and up to 20 seconds with the FLD classifier on data with 10 features, on an i7 processor running at 2.8GHz with 8GB of main memory. In the context of [GLN13], the model is assumed to be private to the client, and hence $\boldsymbol{w}$ is only available in encrypted form to the server. The inner-product $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ needs to be computed with nonscalar multiplications, degrading somewhat the computational performance when compared to the setting considered in this paper. A second source for the poor performance of [GLN13] is related with the chosen number representation. Instead of making use of the canonical embedding to represent real values as in [CKKS17], rational values are scaled by a common factor so that they can be represented as integers, and are

afterwards mapped to a ring $\mathcal{R}_t$:

$$z = \text{sign}(z)(z_s, z_{s-1}, \ldots, z_1, z_0)_2 \rightarrow m_z = \text{sign}(z)(z_0 + z_1 x + \ldots + z_s x^s) \tag{4.4.4}$$

where $(z_s, z_{s-1}, \ldots, z_1, z_0)_2$ corresponds to the binary representation of $|z|$. Homomorphic multiplications and additions respectively multiply and add polynomials in $\mathcal{R}_t$, and the result is evaluated at $x = 2$ to obtain back the corresponding integer. One needs to select a large enough cyclotomic ring $\mathcal{R}$ and a large modulus t, which results in a small multiplicative depth, to ensure that no wrap-around happens. This leads to a significant degradation of performance - although not directly comparable, our reference sequential implementation of classification with an SVM and a linear kernel takes about 3.32s and includes an approximation to the sign value, while [GLN13] takes up to 6s to produce the LM classification, without any approximation of the sign.

4.5 Conclusion

With the recent advances in machine learning, data is becoming increasingly more valuable. When offering classification as a service, companies will be interested in preventing their models' parameters from being disclosed, so that no other provider can mimic their services. Clients are also invested in protecting their own sensitive data. In this context, homomorphic encryption offers theoretically an ideal solution.

While SVMs have shown their effectiveness in classifying data in the past, they are challenging to implement homomorphically, since they require both value-wise and bit-wise arithmetic. In this work we have proposed techniques for homomorphic SVM classification improving both value-wise arithmetic, namely by accelerating polynomial evaluation up to 2.72 times, and bit-wise arithmetic, namely by accelerating sign approximation mechanisms up to 6.59 times. The secrecy of the client's data is ensured by the semantic security of the employed HE scheme, while the leakage of the model parameters is reduced through a randomization of the sign-evaluation method.

While the employed plaintext number representation only allowed for approximate arithmetic, we have shown that it does not affect the model's accuracy. Moreover, the possibility to directly encode up to $n/2$ real values in a single plaintext permit to reach better performances than with the classical batching technique (cf. Section 1.3) where the number of slots available is much smaller than $n/2$.

Future directions of research to proceed with this work include expanding on techniques to evaluate nontrivial functions homomorphically in an efficient manner but also drastically reducing the size of the different keys needed by the server to process the data.

This work is currently in the process of submission of an international conference.

Conclusion and possible future works

Properties of Homomorphic Encryption could be a major asset for privacy protection in the current numeric era. Gentry's breakthrough in 2009 proved that it was achievable and since then a lot of effort has been devoted to make it practical. Despite considerable improvements since the first construction, homomorphic encryption remained hardly practical at the beginning of this thesis and was still limited by its requirements, either in time or in memory. In this thesis we have focused on enhancing performance of the arithmetic used by such encryption schemes in order to contribute at breaking the performance barrier.

An alternative to the costly bootstrapping procedure of Gentry is to use a Somewhat Homomorphic scheme with parameters large enough to evaluate homomorphically the functions required by a targeted application. In this setting, schemes like `BGV` require a chain of moduli $q_0 < \ldots < q_L$ to be able to scale the noise down after each multiplication by switching the ciphertext to a smaller modulus. When evaluating circuits with large multiplicative depth one needs to choose a large chain of moduli and thus use higher dimensions resulting in poor performance. Scale invariant schemes like `FV` allow to partially overcome this limitation by removing the need of the modulus-switching procedure which potentially results in the possibility of evaluating circuits with a bigger multiplicative depth. However, the complex computations required by the procedures of scale-invariant schemes do not allow to use directly the efficient RNS representation all along the computations which somehow cancels the initial benefit brought by these schemes. Our first contribution has been to overcome the natural limitations of the RNS representation to be able to use it during the entire procedures of scale-invariant schemes like `FV` ([BEHZ17]). The significant gains we have obtained by avoiding the reconstruction to a positional representation have permitted to reduce the performance gap with schemes like `BGV`. The methods we have proposed have recently been incorporated in the SEAL library developed by Microsoft Research.

Even though one is able to perform additions and multiplications of plaintext values with homomorphic encryption, the representation of numbers induced by the plaintext space makes the plaintext arithmetic challenging unless encrypting only one bit in each ciphertext. However in this case the ratio between the size of the ciphertext and the size of the plaintext would be huge limiting a little bit more the practicality of homomorphic encryption. The batching technique permits to overcome these difficulties by allowing to pack and process several bits

independently in a single plaintext. Sadly this technique is incompatible with the efficient power-of-two cyclotomics and thus one has to consider other rings with a less efficient arithmetic. Our second contribution has been to improve the arithmetic performance in general cyclotomic rings by focusing on the polynomial reduction procedure required in such cases ([BEH+18]). We have also shown that the impact of our reduction procedures on the noise growth in the `BGV` and `FV` schemes could be handled so that it does not affect the multiplicative depth of these schemes. Despite its considerable impact on the noise growth of scale-invariant schemes such as `FV`, our generic Montgomery reduction is compatible with `BGV` like schemes and could probably benefit a library such as `HElib` in the case of non-power-of-two cyclotomics.

In the big data era, machine-learning offers a concrete solution to solve different decisional problems and in particular for the classification of data. Fully homomorphic encryption allows in theory to perform this classification on encrypted data ensuring therefore their privacy. However this kind of computations require to use a threshold function in the end to return the result. In practice, this threshold function is challenging to implement homomorphically, essentially because it requires comparisons which are hardly compatible with homomorphic operations. In a third contribution we have focused on performing the computations required by a Support Vector Machine (SVM) efficiently ([BMSZ18]). Besides the evaluation of the threshold function, the methods we have proposed to perform the computations required by SVMs are more efficient than some methods from the related art and can fully exploit the strong parallelization potential of devices such as GPU. The experiments we have run show that we are not so far from practical execution time anymore, however the communication costs, and in particular the size of the keys one needs to send to the server prior starting the classification, are still much too important.

Considering that the construction of Gentry in 2009 was not even implementable, we must admit that homomorphic encryption performance have been considerably improved over the last years. Even though we cannot claim yet it is practical enough for an imminent widespread deployment, this idea is no longer completely unrealistic. Of course there is still a lot of work to be done, for instance one should at least gain an additional level of magnitude in performance before even considering such deployment. Since the cryptographic community is still devoting a lot of efforts in improving performance of homomorphic schemes, it is very likely that homomorphic encryption will reach realistic execution time in the near future. However there are others bottlenecks to consider; starting with the communication costs which are still very high. Even though the size of the ciphertexts is not completely unreasonable, the relinearization key and especially the key-switching keys needed by the entity performing the computations can easily require several Giga Bytes of memory. However this problem starts to be considered and recent work of Halevi and Shoup have shown we could significantly reduce the size of these keys (33%-50%) ([HS18]). However this is probably not sufficient yet and we believe this problem would deserve more attention in the upcoming years. Moreover the recent trend for machine-learning has highlighted once more the difficulties to

perform non-trivial computations with homomorphic encryption. Although this problem has already received a lot of attention, the current solutions are not completely satisfactory and would require additional investigations. It is not clear yet, whether or not we will obtain completely generic fully homomorphic encryption, even though big companies like IBM or Microsoft are currently trying to develop efficient and generic libraries for it. However it seems almost certain that we will be able to use efficient homomorphic encryption for several specific applications. As a consequence, identifying these applications and finding efficient ways to perform homomorphically the computations they would require should be a subject of interest for future researches.

List of publications

1. A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes,
   with Jean-Claude Bajard, Julien Eynard and Anwar Hasan (*SAC 2016*).
2. Prover Efficient Public Verification of Dense or Sparse/Structured Matrix-Vector Multiplication,
   with Jean-Guillaume Dumas (*ACISP 2017*).
3. Efficient Reductions in Cyclotomic Rings – Application to Ring-LWE based FHE schemes,
   with Jean-Claude Bajard, Julien Eynard, Paulo Martins, Anwar Hasan and Leonel Sousa (*SAC 2017*).
4. Classification of Private Data using Homomorphic Encryption,
   with Jean-Claude Bajard, Paulo Martins and Leonel Sousa (*in submission process*).

Bibliography

[ABD16] Martin Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and Graded Encoding Schemes. *IACR Cryptology ePrint Archive*, 2016:127, 2016.

[Ajt96] M. Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA, 1996. ACM.

[Ajt98] Miklós Ajtai. The Shortest Vector Problem in L2 is NP-hard for Randomized Reductions (Extended Abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 10–19, New York, NY, USA, 1998. ACM.

[AMBG+16] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint. *Topics in Cryptology - CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, chapter NFLlib: NTT-Based Fast Lattice Library, pages 341–356. Springer International Publishing, Cham, 2016.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9:169–203, October 2015.

[Bar86] P. Barrett. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 1986.

[Bat15] Scott C. Batson. *On the Relationship Between Two Embeddings of Ideals into Geometric Space and the Shortest Vector Problem in Principal Ideal Lattices.* PhD thesis, North Carolina State University, 2015.

[BCIV17] Joppe W. Bos, Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Privacy-friendly forecasting for the smart grid using homomorphic encryption and the group method of data handling. In Marc Joye and Abderrahmane Nitaj, editors, *Progress in Cryptology - AFRICACRYPT 2017*, pages 184–201, Cham, 2017. Springer International Publishing.

[BCN+14] Battista Biggio, Igino Corona, Blaine Nelson, BenjaminI.P. Rubinstein, Davide Maiorca, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. *Security Evaluation of Support Vector Machines in Adversarial Environments*, pages 105–153. Springer International Publishing, 2014.

[BDEM06] Jean-Claude Bajard, Sylvain Duquesne, Milos Ercegovac, and Nicolas Meloni. Residue systems efficiency for modular products summation: Application to Elliptic Curves Cryptography. In *Proceedings of SPIE : Advanced Signal Processing Algorithms, Architectures, and Implementations XVI*, volume 6313, page 0, August 2006.

[BDF17] Guillaume Bonnoron, Léo Ducas, and Max Fillinger. Large FHE gates from Tensored Homomorphic Accumulator. Cryptology ePrint Archive, Report 2017/996, 2017. `https://eprint.iacr.org/2017/996`.

[BEH+18] Jean-Claude Bajard, Julien Eynard, Anwar Hasan, Paulo Martins, Leonel Sousa, and Vincent Zucca. Efficient Reductions in Cyclotomic Rings - Application to Ring-LWE Based FHE Schemes. In Carlisle Adams and Jan Camenisch, editors, *Selected Areas in Cryptography – SAC 2017*, pages 151–171, Cham, 2018. Springer International Publishing.

[BEHZ17] Jean-Claude Bajard, Julien Eynard, M. Anwar Hasan, and Vincent Zucca. A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes. In Roberto Avanzi and Howard Heys, editors, *Selected Areas in Cryptography – SAC 2016*, pages 423–442, Cham, 2017. Springer International Publishing.

[BEMP15] J.-C. Bajard, J. Eynard, N. Merkiche, and T. Plantard. RNS Arithmetic Approach in Lattice-Based Cryptography: Accelerating the "Rounding-off" Core Procedure. In *Computer Arithmetic (ARITH), 2015 IEEE 22nd Symposium on*, pages 113–120, June 2015.

[BF16] Guillaume Bonnoron and Caroline Fontaine. A note on Ring-LWE security in the case of Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive*, 2016:385, 2016.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In Joe Kilian, editor, *Theory of Cryptography*, pages 325–341, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[BI04] J. Bajard and L. Imbert. A Full RNS Implementation of RSA. *Computers, IEEE Transactions on*, 53(6):769–774, June 2004.

[BIN06] Jean-Claude Bajard, Laurent Imbert, and Christophe Negre. Arithmetic Operations in Finite Fields of Medium Prime Characteristic Using the Lagrange Representation. *IEEE Transactions on Computers*, 55:1167–1177, 2006.

[BJ18] Jim Basilakis and Bahman Javadi. Efficient Parallel Binary Operations on Homomorphic Encrypted Real Numbers. Cryptology ePrint Archive, Report 2018/201, 2018. `https://eprint.iacr.org/2018/201`.

[BLLN13] JoppeW. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In Martijn Stam, editor, *Cryptography and Coding*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer Berlin Heidelberg, 2013.

[BMSZ18] Jean-Claude Bajard, Paulo Martins, Leonel Sousa, and Vincent Zucca. Classification of Private Data using Homomorphic Encryption. currently in the process of submission, 2018.

[Bra12] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 868–886. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 97–106, Washington, DC, USA, 2011. IEEE Computer Society.

[BV18] Charlotte Bonte and Frederik Vercauteren. Privacy-Preserving Logistic Regression Training. Cryptology ePrint Archive, Report 2018/233, 2018.

[CDSM15] Gizem S Cetin, Yarkin Doroz, Berk Sunar, and William J Martin. Arithmetic Using Word-Wise Homomorphic Encryption. Cryptology ePrint Archive, Report 2015/1195, 2015.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 3–33, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[CGH+18] Jack L.H. Crawford, Craig Gentry, Shai Halevi, Daniel Platt, and Victor Shoup. Doing Real Work with FHE: The Case of Logistic Regression. Cryptology ePrint Archive, Report 2018/202, 2018. `https://eprint.iacr.org/2018/202`.

[Che16] Cheon, Jung Hee and Kim, Andrey and Kim, Miran and Song, Yongsoo. Implementation of hea-an, 2016.

[CHH+17] Hao Chen, Kyoohyung Han, Zhicong Huang, Amir Jalali, and Kim Laine. *Simple Encrypted Arithmetic Library v2.3.0*. Microsoft Research, December 2017.

[CHK+18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for Approximate Homomorphic Encryption. Cryptology ePrint Archive, Report 2018/153, 2018. `https://eprint.iacr.org/2018/153`.

[CIV16] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. On Error Distributions in Ring-based LWE. *LMS Journal of Computation and Mathematics*, 19(A):130–145, 2016.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1—-27:27, 2011.

[CLP17] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library - seal v2.1. In *Financial Cryptography and Data Security: FC 2017 International Workshops,*

WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers, pages 3–18, Cham, 2017. Springer International Publishing.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better Lattice Security Estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 1–20, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[CP15] Eric Crockett and Chris Peikert. $\Lambda \circ \lambda$: Functional Lattice Cryptography. Cryptology ePrint Archive, Report 2015/1134, 2015. `http://eprint.iacr.org/2015/1134`.

[CSVW17] Anamaria Costache, Nigel P. Smart, Srinivas Vivek, and Adrian Waller. Fixed-Point Arithmetic in SHE Schemes. In Roberto Avanzi and Howard Heys, editors, *Selected Areas in Cryptography – SAC 2016*, pages 401–422, Cham, 2017. Springer International Publishing.

[CT65] James Cooley and John Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[DD12] Léo Ducas and Alain Durmus. Ring-LWE in Polynomial Rings. In *Public Key Cryptography – PKC 2012: 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, pages 34–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[DF04] D.S. Dummit and R.M. Foote. *Abstract Algebra*. Wiley, 2004.

[DH76] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 1976.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 617–640, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *Proceedings of the 32Nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, pages 643–662, New York, NY, USA, 2012. Springer-Verlag New York, Inc.

[DS16] Wei Dai and Berk Sunar. *cuHE: A Homomorphic Encryption Accelerator Library*, pages 169–186. Springer International Publishing, Cham, 2016.

[ElG85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 10–18, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[Erd46] Paul Erdös. On the coefficients of the cyclotomic polynomial. *Bull. Amer. Math. Soc.*, 52(2):179–184, 02 1946.

[Fan] Rong-En Fan. LIBSVM Data: Classification, Regression, and Multi-label.

[Fil00] Michael Filaseta. On coverings of the integers associated with an irreducibility theorem of A. Schinzel. Number theory for the millennium, II (Urbana, IL, 2000), A K Peters, Natick, MA, 2002, 1-24., 2000.

[FP97] W. L. Freking and K. K. Parhi. Low-power FIR digital filters using residue arithmetic. In *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No.97CB36136)*, volume 1, pages 739–743 vol.1, Nov 1997.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive*, 2012.

[Gar59] Harvey L. Garner. The Residue Number System. In *Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference*, IRE-AIEE-ACM '59 (Western), pages 146–153, New York, NY, USA, 1959. ACM.

[GDL+16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 201–210. JMLR.org, 2016.

[Gen09] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[GH11] Craig Gentry and Shai Halevi. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 129–148, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully Homomorphic Encryption with Polylog Overhead. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 465–482, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. *Homomorphic Evaluation of the AES Circuit*, pages 850–867. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[GLN13] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML Confidential: Machine Learning on Encrypted Data. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *Information Security and Cryptology – ICISC 2012*, pages 1–21, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting Lattice Reduction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 31–51, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75–92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Gt15] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6.1.0 edition, 2015. `http://gmplib.org/`.

[Har14] David Harvey. Faster Arithmetic for Number-Theoretic Transforms. *J. Symb. Comput.*, 60:113–119, January 2014.

[HHSSD17] Shai Halevi, Tzipora Halevi, Victor Shoup, and Noah Stephens-Davidowitz. Implementing BP-Obfuscation Using Graph-Induced Encoding. Cryptology ePrint Archive, Report 2017/104, 2017. http://eprint.iacr.org/2017/104.

[HPS18] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An Improved RNS Variant of the BFV Homomorphic Encryption Scheme. Cryptology ePrint Archive, Report 2018/117, 2018. https://eprint.iacr.org/2018/117.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In *CRYPTO*, pages 554–571. Springer, 2014.

[HS18] Shai Halevi and Victor Shoup. Faster Homomorphic Linear Transformations in HElib. Cryptology ePrint Archive, Report 2018/244, 2018. https://eprint.iacr.org/2018/244.

[Isa94] I.M. Isaacs. *Algebra: A Graduate Course*. Graduate studies in mathematics. American Mathematical Society, 1994.

[KGV16] A. Khedr, G. Gulak, and V. Vaikuntanathan. SHIELD: Scalable Homomorphic Implementation of Encrypted Data-Classifiers. *IEEE Transactions on Computers*, 65(9):2848–2858, Sept 2016.

[KSW+18] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang. Secure Logistic Regression based on Homomorphic Encryption. Cryptology ePrint Archive, Report 2018/074, 2018.

[Lan05] S. Lang. *Algebra*. Graduate Texts in Mathematics. Springer New York, 2005.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *MATH. ANN*, 261:515–534, 1982.

[LLM06] Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. Cryptographically private support vector machines. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, page 618, New York, New York, USA, 2006. ACM Press.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized Compact Knapsacks Are Collision Resistant. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 144–155, Berlin, Heidelberg, 2006. Springer-Verlag.

[LMPR08] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A Modest Proposal for FFT Hashing. In Kaisa Nyberg, editor, *Fast Software Encryption: 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, pages 54–72, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes fv and yashe. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology – AFRICACRYPT 2014: 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, pages 318–335, Cham, 2014. Springer International Publishing.

[LN16] Patrick Longa and Michael Naehrig. *Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography*, pages 124–139. Springer International Publishing, Cham, 2016.

[LP11] Richard Lindner and Chris Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 319–339, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[LP16] Kim Laine and Rachel Player. Simple encrypted arithmetic library - seal (v2.0). Technical report, Microsoft, September 2016.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *On Ideal Lattices and Learning with Errors over Rings*, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *A Toolkit for Ring-LWE Cryptography*, pages 35–54. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[LS12] Adeline Langlois and Damien Stehlé. Hardness of decision (R)LWE for any modulus. *IACR Cryptology ePrint Archive*, 2012:91, 2012. informal publication.

[Lyu11] V. Lyubashevsky. Search to decision reduction for the learning with errors over rings problem. In *2011 IEEE Information Theory Workshop*, pages 410–414, Oct 2011.

[Mer78] Ralph C. Merkle. Secure Communications over Insecure Channels. *Commun. ACM*, 21(4):294–299, April 1978.

[Mic01] Daniele Micciancio. The Shortest Vector Problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, March 2001. Preliminary version in FOCS 1998.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[MS17] Paulo Martins and Leonel Sousa. Enhancing data parallelism of fully homomorphic encryption. In Seokhie Hong and Jong Hwan Park, editors, *Information Security and Cryptology – ICISC 2016: 19th International Conference, Seoul, South Korea, November 30 – December 2, 2016, Revised Selected Papers*, pages 194–207, Cham, 2017. Springer International Publishing.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[MW16] Daniele Micciancio and Michael Walter. Practical, Predictable Lattice Basis Reduction. In *Proceedings, Part I, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9665*, pages 820–849, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM.

[Pai99] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[Pei16] Chris Peikert. How Not to Instantiate Ring-LWE. In *Proceedings of the 10th International Conference on Security and Cryptography for Networks - Volume 9841*, pages 411–430, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[POG15] Thomas Pöppelmann, Tobias Oder, and Tim Güneysu. High-Performance Ideal Lattice-Based Cryptography on 8-Bit ATxmega Microcontrollers. In *Proceedings of the 4th International Conference on Progress in Cryptology – LATINCRYPT 2015 - Volume 9230*, pages 346–365, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

[PRSD17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In *STOC 2017 - Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, volume Part F128415, pages 461–473. Association for Computing Machinery, 6 2017.

[PS73] Michael S. Paterson and Larry J. Stockmeyer. On the Number of Nonscalar Multiplications Necessary to Evaluate Polynomials. *SIAM Journal on Computing*, 2(1):60–66, mar 1973.

[QWD+16] Junfei Qiu, Qihui Wu, Guoru Ding, Yuhua Xu, and Shuo Feng. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1):67, dec 2016.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[Reg05] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[Rem34] E Remez. Sur la détermination des polynômes d'approximation de degré donnée. *Communications of the Kharkov Mathematical Society*, 10:41–63, 1934.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[RSR69] Lawrence R. Rabiner, Ronald W. Schafer, and Charles M. Rader. The Chirp z-Transform Algorithm and Its Application. *Bell System Technical Journal*, 48(5):1249–1292, 1969.

[RVM+14] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact Ring-LWE Cryptoprocessor. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 371–391, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[SE94] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1):181–199, Aug 1994.

[SGS10] J E Stone, D Gohara, and G Shi. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. *IEEE Des. Test*, 12(3):66–73, 2010.

[Sha49] C. Shannon. Communication Theory of Secrecy Systems. *Bell Systems Techn. Journal*, 28:656–719, 1949.

[Sho97] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997.

[SK89] A. P. Shenoy and R. Kumaresan. Fast Base Extension Using a Redundant Modulus in RNS. *IEEE Trans. Comput.*, 38(2):292–297, February 1989.

[ST67] Nicholas S. Szabo and Richard I. Tanaka. *Residue Arithmetic and Its Applications to Computer Technology*. McGraw-Hill Book Company, New York, 1967.

[SV14] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.

[vdPS13] Joop van de Pol and Nigel P. Smart. Estimating Key Sizes for High Dimensional Lattice-Based Systems. In Martijn Stam, editor, *Cryptography and Coding: 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, pages 290–303, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[vEB81] P. van Emde-Boas. *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*. Report. Department of Mathematics. University of Amsterdam. Department, Univ., 1981.

[Was97] L.C. Washington. *Introduction to Cyclotomic Fields*. Graduate Texts in Mathematics. Springer New York, 1997.