A journey towards practical fully homomorphic encryption

Guillaume Bonnoron

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE
COMUE UNIVERSITE BRETAGNE LOIRE

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

Par

Guillaume BONNORON

A journey towards practical Fully Homomorphic Encryption

Thèse présentée et soutenue à à IMT Atlantique, le 15 mars 2018
Unité de recherche : Lab-STICC (UMR CNRS 6285) et Chaire de cyberdéfense des systèmes navals
Thèse N° : 2018IMTA0073

Composition du Jury :

Président :	Damien STEHLE	Professeur, LIP, Ecole Normale Supérieure de Lyon
Rapporteurs :	Frederik VERCAUTEREN	Associate Professor, KU Leuven, Belgique
	Thomas JOHANSSON	Professor, Lund University, Suède
Examinateurs	Renaud SIRDEY	Directeur de Recherche, LIST, CEA Saclay
	Léo DUCAS	Chercheur, CWI, Amsterdam, Pays-Bas
Dir. de thèse :	Caroline FONTAINE	Chargée de recherche CNRS, IMT Atlantique, Lab-STICC

Invités

Adeline ROUX-LANGLOIS	Chargée de Recherche CNRS, IRISA, Rennes
Sylvain LACHARTRE	Chercheur, Thales Com&Security, Gennevilliers

Contents

Thanks

Mes premiers remerciements vont à ma directrice de thèse, Caroline. La genèse de cette thèse nous ramène en 2013 (et même avant), aussi je la remercie immensément de m'avoir permis d'atteindre cet objectif, grâce notamment à tout son soutien et ses encouragements.

Merci beaucoup à Sylvain et Jacques et Vincent de Thales d'avoir aiguillé les réalisations de cette thèse.

To my reviewers, Frederik and Thomas, I am most grateful for the time they have spent, reading my prosis and providing their great feedbacks. Merci aux examinateurs et invités, Damien, Léo, Adeline, Renaud, Sylvain, Caroline de m'avoir permis de restituer ces trois années de travail.

Comme j'ai pu le constater, l'environnement dans lequel se déroule une thèse est très important. Aussi je salue et je remercie grandement toute la troupe de la Caverne : Benjamin, Thibaud, Bastien, David, Yvon, Xavier, Olivier, Étienne et Arthur, et les personnes qui y sont passées : Pedro, Thomas, Gaël, Erwan, Alex, Guillaume. Merci à Philippe, Patrick, Caroline, David et Yvon pour l'animation et le cadre qu'ils assurent au sein de notre chaire. Pour leur patience et leur support, un grand merci à Magalie, Christine et Amélie, sans qui je n'aurai jamais pu rajouter cette dimension collaborative et m'intégrer si bien dans la communauté.

Big thanks to my co-authors who with I have always had smooth and fruitful collaboration : Caroline, Vincent, Léo, Max, Damien, Teja, Guillaume.

To my fellow `fplll` development team members: Martin, Marc, Damien, Léo, Shi, Mickael, Koen, a big thanks for the interaction we have had during those coding days in Lyon, interaction that reached beyond lines of codes.

For making my stay there possible, I send my warmest thanks to the CWI staff. Many thanks to Léo for the invitation, to Max for the collaboration and to the whole crypto team: Cécile, Pierre, Marc, Serge, Koen, Ronald and the others I forget!

Merci aussi à tous les cryptos français avec qui j'ai eu la chance de discuter au cours de ces trois ans : Julien, Vincent, Soukayna, Adeline, Thomas, Fabrice, Marie, Thierry, Pierre-Alain, Renaud, Malika, Phong, Pauline, Chen, Rachel, Kim, Pascal.

Et puisqu'une thèse ce n'est pas que de la recherche scientifique, je voudrais saluer tous ceux avec qui j'ai partagé des discussions sport, nutrition et bien d'autres encore. La principale question non élucidée reste de savoir si la thèse c'est paléo !

Enfin mes pensées vont à mes proches sans qui je ne serai, bien sûr, jamais arrivé là. Je pense à mes parents, mes frère et sœur et tout particulièremnt à ma tendre Lavi avec qui j'ai le plaisir de partager ma vie.

Abstract

Fully Homomorphic Encryption has been around since 2009 and the seminal work of Gentry, nearly thirty years after the concept was imagined. This kind of encryption is that which allows to compute on encrypted data, with the guarantee that the outcomes of the computation, once decrypted, will be the same as if the computation had been done on the un-encrypted data. Since Gentry's work, the community has been tremendously dynamic on this topic and we have seen many scheme proposals which have kept improving the prior works.

The aim of the thesis is to study the gap that remains between the theoretical proposals and the use in real life applications of homomorphic encryption. Our contribution is divided in several topics: the study of the concrete security of the proposals, the comparison of the most promising schemes on a wide range of use cases and the conception and implementation of a new scheme.

To begin, we focus our attention to the cryptanalysis aspects. The security of the homomorphic schemes stands on the ground that the Learning With Errors (LWE) problem is hard. However, for real use we need more than asymptotic hardness result. We want to setup the scheme so that we guarantee say 80 or 128 bits of security. Due to the relative youth of the LWE problem, at cryptographic scale, we still lack hindsight on its concrete security. As our first task, we studied the practical security of LWE, in the cases of homomorphic encryption. We derived a special-purpose attack that we implemented. Our work improved upon the state-of-the-art at that time and our attack showed unexpectedly good performance for some cases.

Once we had gained confidence on how to setup a homomorphic scheme correctly, we could go on to our next task: comparing the existing schemes. Indeed, the details of the different proposals are very technical and for people from outside the cryptographic community (e.g. implementers) it is very hard to know which scheme is the most appropriate. So, we join effort with Vincent Migliore to conduct a comparative survey of the most promising schemes as of 2016. Our aim was to provide a most comprehensive study on different use cases, under different implementation constraints so that we could say which scheme stands out for which situation, and so that implementers could find ready-to-use information.

Last line of work in this thesis, the conception and implementation of a new scheme. I visited at the CWI in Amsterdam in Spring 2017 and worked with Léo Ducas and Max Fillinger on an improvement of FHEW. Our new proposal belongs to the generation of the bootstrapped schemes, that are those where the noise is kept constant between the gates. With our proof-of-concept implementation we can evaluate a binary gate on six input bits

with one output bit in roughly six seconds.

Since 2015, we and the community have moved a long way towards practical homomorphic encryption. With the recent performance and the maturity that has been gained, we can expect homomorphic encryption use in real world application within five to ten years.

Résumé en français

Introduction

Contexte

Cette thèse s'inscrit dans le cadre de la Chaire de cyberdéfense des systèmes navals[1]. Fondée en 2014 autour de deux partenaires académiques Télécom Bretagne (aujourd'hui IMT Atlantique) et l'École Navale ainsi que deux partenaires industriels Thales et DCNS (aujourd'hui Naval Group), elle a pour objectif d'organiser un effort de recherche sur les problèmes de cyberdéfense dans l'environnement naval, d'abord militaire mais aussi civil. Nous sommes plusieurs doctorants à travailler sur des sujets de cyberprotection, cyberdéfense et cyberrésilience. Mon sujet plus particulièrement s'intéresse à la protection des données.

Organisation

L'objet de cette thèse est d'étudier dans quelles mesures la cryptographie homomorphe pourrait être utilisée à bord des navires. Celle-ci permettant l'utilisation de données chiffrées dans un programme, cela consisterait en un obstacle supplémentaire pour un attaquant souhaitant voler des données opérationnelles. Étant donné le dynamisme de ce sujet au sein de la communauté cryptographique, nous avons focalisé notre attention sur trois sujets principaux, qui seront repris dans leur ordre de traitement chronologique dans ce manuscrit.

- D'abord, nous avons endossé un rôle d'attaquant afin d'avoir une bonne compréhension des façons de casser cette cryptographie. Cela nous a conduit à imaginer une nouvelle attaque, reposant sur des résultats existants, mais dans un cadre plus ciblé permettant d'améliorer l'état de l'art. Ce résultat a fait l'objet d'une publication à IndoCrypt 2017 (chapitre 3). Ce travail nous a permis de rejoindre la communauté de cryptanalyse algorithmique autour notamment du développement de la bibliothèque de fonctions `fplll` (chapitre 4).

- Avec la maîtrise du volet attaque, nous avons donc décidé de faire un état de l'art des schémas homomorphes existants et de les comparer. Ce travail, effectué avec Vincent Migliore, a vocation à faciliter le travail aux implémenteurs, pas nécessairement cryptographes, en leur fournissant notamment des tables de paramètres (sûrs et corrects) précalculées, adressant de multiples cas d'usages (chapitre 5).

[1] `https://www.ecole-navale.fr/Chaire-de-cyberdefense-des.html`

- Enfin, lors d'un stage doctoral au CWI à Amsterdam, nous avons travaillé avec Léo Ducas et Max Fillinger à la construction et implémentation d'un nouveau schéma de chiffrement homomorphe, généralisant un schéma existant en lui apportant de multiples améliorations (chapitre 6).

Nous invitons le lecteur à parcourir le chapitre 2 pour les quelques notations et définitions qui pourraient être nécessaires à la lecture de ce résumé.

Présentation de la cryptographie homomorphe

Imaginée en 1978 [RAD78], la cryptographie *complètement* homomorphe n'a été réalisée pour la première fois qu'en 2009 par les travaux de Gentry [Gen09]. Cette cryptographie a pour but de permettre la réalisation de traitement sur des données chiffrées de telle façon que le résultat puisse être déchiffré et soit identique au résultat qui aurait été obtenu si ce traitement avait eu lieu sur les mêmes données non chiffrées. C'est le caractère *homomorphe* du traitement.

Depuis 2009, de nombreux schémas homomorphes ont vu le jour, améliorant les performances ou réduisant les hypothèses nécessaires à leurs sécurités. Une première famille de schémas repose sur l'utilisation d'entiers [vDGHV10, CMNT11, CNT12], une autre, plus prometteuse, sur des réseaux euclidiens (*lattices* en anglais) [BV11b, BGV12, FV12, BLLN13]. Tous ces schémas constituent la deuxième génération. C'est la plus mature et ses performances sont très intéressantes, nous avons donc commencé notre travail avec ses schémas. Ensuite [GSW13, KGV16] ont introduit une troisième génération de schémas et enfin, les propositions les plus récentes apparues au cours de cette thèse [DM15, CGGI16, CGGI17] forment ce que nous pouvons appeler la quatrième génération. Notre travail au CWI s'inclut dans cette dernière vague.

Un exemple : le schéma de Fan-Vercauteren

Introduit en 2012 comme une généralisation sur anneaux du schéma [Bra12], le schéma de Fan et Vercauteren [FV12], que nous appellerons FV, a rencontré beaucoup de succès. C'est notamment le schéma choisi par Microsoft Research pour leur bibliothèque de fonctions [LCP17]. On retrouve également une implémentation de FV sur NFLlib [Cry]. Simple à énoncer et reprenant les principaux éléments de sa génération, c'est ce schéma qui nous a servi comme porte d'entrée, pour notre travail de cryptanalyse notamment.

Le schéma. Pour la description de FV nous avons besoin d'un anneau R de dimension n. Pour une exposition facile, le choix est souvent fait de définir $R = \mathbb{Z}[x]/(f(x))$ où $f(x) = x^n + 1$ est un polynôme cyclotomique et donc n une puissance 2. De plus nous restreignons à R_q, l'anneau composé des éléments de R aux coefficients dans $]-q/2, q/2]$ pour un module q fixé.

- Génération de clés : La clé secrète s_k est un élément de R_q, généralement petit. La clé publique est $p_k \in R_q^2$ définie par :

$$p_k = (\mathbf{p_0}, \mathbf{p_1}) = (-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) \mod q, \mathbf{a})$$

où $\mathbf{s}$ est la clé secrète, $\mathbf{a}$ et $\mathbf{e}$ sont tirés aléatoirement, uniformément dans R_q pour $\mathbf{a}$ et selon une gaussienne discrète d'écart type σ dans R pour $\mathbf{e}$.

- Chiffrement : l'ensemble des clairs est l'anneau R_t pour un entier $t \geq 2$. Le chiffrement d'un message $\mathbf{m} \in R_t$ se réalise comme suit :

$$\mathbf{c} = (\mathbf{c_0}, \mathbf{c_1}) = (\lfloor q/t \rfloor \cdot \mathbf{m} + \mathbf{p_0} \cdot \mathbf{u} + \mathbf{e_1}, \mathbf{p_1} \cdot \mathbf{u} + \mathbf{e_2})$$

  où $\mathbf{u}$ est tiré aléatoirement uniforme dans R_2 et $\mathbf{e_1}, \mathbf{e_2}$ sont tirés selon une gaussienne d'écart type σ dans R.

- Pour déchiffrer, il suffit de calculer $\lfloor (\mathbf{c_0} + \mathbf{c_1} \cdot \mathbf{s}) \times t/q \rceil$ pour retrouver $\mathbf{m}$.

La partie intéressante est donc comment les opérations se déroulent sur les chiffrés. Donnons nous $\mathbf{c}^a$ et $\mathbf{c}^b$ des chiffrés de $\mathbf{m}^a$ et $\mathbf{m}^b$.

- L'addition se fait simplement terme à terme

$$\mathbf{c}^+ = (\mathbf{c_0}^+, \mathbf{c_1}^+) = (\mathbf{c_0}^a + \mathbf{c_0}^b, \mathbf{c_1}^a + \mathbf{c_1}^b) \in R_q^2$$

  Il est aisé de vérifier qu'il s'agit bien d'un chiffré de $\mathbf{m}^a + \mathbf{m}^b$.

- La multiplication est un peu plus complexe puisque l'analogue à l'addition terme à terme, fait apparaître une troisième terme. Il est donc nécessaire de réaliser une opération de *relinéarisation* afin de retrouver un chiffré valide. Cette relinéarisation, nécessite du matériel de clés supplémentaire, en plus de la clé publique, que l'on nomme simplement clé de relinéarisation.

Grâce à cette brève présentation, on peut noter que FV est paramétré par différentes variables : l'anneau R, sa dimension n, un module q et un écart type pour l'erreur σ. Il en est de même pour la plupart des schémas homomorphes et même plus largement de la cryptographie à base de réseaux euclidiens.

Conception d'une nouvelle attaque

Notre premier travail de recherche a consisté à faire l'état des lieux des attaques possibles contre les schémas de chiffrement homomorphe afin de déduire par la suite des paramètres sûrs pour une utilisation maîtrisée. Le problème difficile principal sur lequel repose la plupart des solutions cryptographiques sur les réseaux euclidiens est le problème *Learning with Errors* ou LWE [Reg05]. Celui a pour but d'apprendre un secret masqué au milieu d'équations linéaires bruitées. Plus précisément, étant donné une dimension n, un module q et une distribution de probabilités χ d'écart type σ, fixons $\mathbf{s} = (s_1, ..., s_n)$ et construisons m couples de la façon suivante :

$$(\mathbf{a_i}, \langle \mathbf{a_i} \cdot \mathbf{s} \rangle + \mathbf{e_i} \mod q)$$

où chaque $\mathbf{a_i}$ et $\mathbf{e_i}$ sont constitués de n coordonnées tirées uniformément aléatoires pour $\mathbf{a_i}$ et selon la distribution de probabilités χ pour $\mathbf{e_i}$. Ces m échantillons sont rendus publics et deux

problèmes en découlent : Decision-LWE (décider si ces m échantillons ont été construits ainsi ou de façon complètement aléatoire) et Search-LWE (retrouver $\mathbf{s}$). Un problème similaire a été défini avec des anneaux, Ring-LWE [LPR10] pour lequel il existe également une variante décisionnelle et une variante recherche. Il a été démontré de nombreux résultats garantissant que ces problèmes sont difficiles asymptotiquement. Il reste donc à estimer leurs difficultés concrètes étant donné un jeu de paramètres (n, q, σ) fixé. Pour cela, il convient d'analyser les performances des différentes méthodes de résolution de ces problèmes.

Juste avant le début de cette thèse, Albrecht et al. ont réalisé une excellente synthèse pour LWE [APS15]. Plus tard Peikert en a également réalisé une pour Ring-LWE [Pei16]. En nous appuyant sur ces travaux, nous avons pu nous intéresser de plus près aux performances de ces attaques sur des instances avec des paramètres propres à la cryptographie homomorphe. En effet toutes ces attaques sont génériques, or comme nous le verrons par la suite en détails, le caractère homomorphe impose des contraintes sur le triplet (n, q, σ) qui pourraient être mises à profit par un attaquant.

La nouvelle attaque

Profitant d'une erreur petite et d'un module trÈs grand, nous avons composé une attaque en suivant le modèle de celle de Bai et Galbraith [BG14]. Ces résultats ont été présentés lors de la conférence IndoCrypt 2017.

Plongement. Notre montage diffère sur quelques points. Le plongement n'est pas réalisé exactement de la même façon. Nous le réalisons de la façon suivante

$$\mathbf{B} = \begin{pmatrix} \mathbf{I_n} & -\mathbf{A} \\ \mathbf{0} & q\mathbf{I_n} \end{pmatrix} \in \mathbb{Z}^{2n \times 2n}$$

Alors que Bai et Galbraith le font comme suit

$$\mathbf{M} = \begin{pmatrix} \mathbf{I_n} & -\mathbf{A} \\ & q\mathbf{I_{2n}} \end{pmatrix} \in \mathbb{Z}^{3n \times 2n}$$

devant ensuite en calculer la forme normale de Hermite pour obtenir une matrice de rang complet. Notre solution est donc sensiblement plus efficace et nous donne une base très particulière (triangulaire supérieure, par bloc, avec l'identité sur la diagonale à un facteur près). Cette forme particulière s'avère en effet utile pour l'étape suivante de l'attaque : la recherche de vecteur le plus proche.

Réduction de réseaux. Une recherche de vecteur le plus proche passe nécessairement par une phase de réduction du réseau concerné afin que la base soit favorable à la recherche effective. Plusieurs algorithmes s'offrent à nous pour cette opération : LLL [LLL82] de faible complexité mais dont la qualité de réduction est moyenne ou BKZ [SE94] moins rapide mais produisant des bases de meilleure qualité. Nous avons analysé lequel de ces algorithmes est le plus pertinent pour notre cas ainsi que les meilleurs paramètres à lui définir. Il s'est avéré qu'un LLL faible, donc très rapide, nous permettait de réussir nos attaques.

Énumeration. Une fois la base réduite, plusieurs algorithmes existent pour réaliser une recherche de vecteur proche. Nous avons opté pour l'énumération [LN13]. Cette méthode est la meilleure pour les instances accessibles aujourd'hui mais nous savons que les méthodes de crible sont plus performantes asymptotiquement et pourraient le devenir aussi en pratique [HK17].

Expériences réelles

Après avoir assemblé de façon théorique les différents éléments de cette attaque, nous en avons réalisé une implémentation pour étudier ses performances pratiques. Notre cible d'attaque a été l'implémentation de Lepoint [Lep14]. Si nous devions faire les mêmes expériences aujourd'hui, notre choix porterait sur la bibliothèque SEAL [LCP17] qui, elle, a vocation à être réellement utilisée.

Pour l'attaque, nous avons utilisé la bibliothèque `fplll` [dt17] pour la réduction des réseaux. Le reste a été implémenté par nos soins. Par la suite nous avons découvert que `fplll` avait également du code « expérimental » pour l'énumération avec lequel nous avons travaillé par la suite.

Résultats. Ces expériences, lancées contre des centaines d'instances de FV avec différents paramètres de dimension, module et erreur nous ont d'abord amenés à considérer la réduction de réseaux. En effet, dans l'idéal les efforts fournis sur la réduction doivent être équilibrés par rapport à ceux fournis sur l'énumération. Or il s'avère qu'avec nos premiers choix (BKZ notamment), la quasi-totalité du temps était passée sur la réduction. Nous avons donc évolué petit à petit vers une réduction de plus en plus faible jusqu'à un faible LLL ($\eta = 0.71$). La partie énumération, quant à elle, resta très rapide également. De façon surprenante, nous n'avions nul besoin d'énumérer puisque le premier candidat (la solution de Babai [Bab86]) était presque toujours celle que nous cherchions.

Nous avions donc entre les mains une attaque composée uniquement de procédures aux coûts polynomiaux en temps. Cela aurait pu être un vrai problème pour la cryptographie homomorphe. Toutefois, après avoir longuement approfondi l'analyse du comportement de notre attaque, nous avons établi que celle-ci ne passerait pas à l'échelle et n'inquiéterait pas des instances proposant 80 bits de sécurité par exemple.

Bien que les retombés de notre travail soit limité, la communauté l'a suivi de très prêt, puisque c'était un des premiers de ce genre. L'an dernier Albrecht [Alb17] a présenté un travail similaire aux résultats plus forts.

Contribution en réduction de réseaux

Du fait de notre investissement sur le sujet de la réduction de réseaux pour la compréhension de notre attaque, nous avons rejoint la communauté gravitant autour du projet `fplll` [dt17], notamment lors de rencontres de programmation : les `fplll` days. Ces évènements

réunissent deux fois par an les intéressés pour faire avancer le projet : ajout de fonctionnalités, optimisation, documentation... Nous avons donc pu apporter notre concours à ce projet à ces occasions.

Améliorer la recherche de proches vecteurs. Réel besoin pour notre attaque, cette fonction que nous avions découverte par hasard n'était pas recommandée à l'usage. Nous avons donc consacré du temps à la rendre correcte. Ces problèmes étaient déjà évoqués par Pujol [Puj08].

BKZ 3. Un grand nombre d'heuristiques ont été proposées pour améliorer les performances de BKZ en pratique. Elles ont été regroupées sous le nom de BKZ 2.0 [CN11]. Ces résultats ayant déjà quelques années, nous avons travaillé à plusieurs lors des journées `fplll` pour tester de nouvelles heuristiques et ouvrir le chemin vers un futur « BKZ 3.0 ». De multiples techniques ont été mises à l'épreuve, certaines déjà intégrées d'autres pas encore. Il est encore trop tôt pour réaliser une publication.

LLL en deux phases. Phénomène déjà constaté auparavant, nous avons pu mettre en évidence lors de nos multiples expériences avec l'algorithme LLL, qu'il était souvent plus rapides de le réaliser en plusieurs passes, avec des paramètres moins exigeants que ceux souhaités en tant que pré-calculs. Il ne s'agit pas d'une vérité universelle mais les gains que nous avons ne sont pas négligeables. Aussi il est intéressant d'en faire l'expérience lors de réduction de réseaux avec ces propriétés.

Choix de l'algorithme et des paramètres

Forts de ces expériences en cryptanalyse, nous avons pu nous tourner avec confiance vers la tâche suivante, celle du choix du meilleur cryptosystème homomorphe et de ses paramètres. En commun avec Vincent Migliore, nous avons donc effectué cette étude comparative dans le but de fournir un contenu d'accès facile pour les implémenteurs. Ce travail a été accepté pour un numéro spécial des Transactions on Computers de l'IEEE sur « l'ingénierie cryptographique dans le contexte post-quantique » [**?**].

Périmètre de l'étude

Ce travail débuté à l'été 2016 nous avons d'abord considéré les schémas les plus prometteurs à cette époque, c'est-à-dire YASHE' [BLLN13], FV [FV12] de deuxième génération ainsi que SHIELD (ou Ring-GSW) [KGV16] et F-NTRU [DS16] de troisième génération. Plus tard, suite à différentes attaques [ABD16, KF17], nous avons dû retirer YASHE' et F-NTRU qui n'était plus viable. Notre étude permet néanmoins de mettre en regard des schémas de deuxième et troisième génération sur des cas très concrets.

Nous nous sommes ensuite donnés plusieurs cas d'usages afin de mettre en évidence les avantages et inconvénients des schémas selon les situations. Il en découle de nombreuses tables de paramètres. Les valeurs des paramètres permettent ensuite de se faire un avis sur le

coût d'utilisation de chacun des schémas. En effet les tailles de clés ou chiffrés en dépendent directement, tout comme les temps de calcul, eux aussi liés à la taille des données manipulées.

Réalisations

Pour réaliser cette étude nous avons donc remis les schémas sous un formalisme commun, afin d'en dériver des équations de fonctionnement comparables. Calculer les paramètres revient donc à trouver les valeurs qui satisfont à la fois ces équations de fonctionnement et les conditions de sécurité.

Pour ce faire, nous avons donc réalisés un script de recherche. Celui-ci inclus les équations de fonctionnement des schémas et s'interface avec l'estimateur de sécurité de Albrecht [Albb]. Celui-ci étant toujours mis à jour à mesure que de nouvelles techniques d'attaques sont publiées il nous paraissait important de pouvoir maintenir nos tables à jour facilement.

Résultats

Chaque situation mérite une attention particulière, néanmoins plusieurs tendances se dégagent de l'ensemble de nos résultats.

1. Pour des petites profondeurs multiplicatives, FV présente de plus petits paramètres, alors que si le circuit à évaluer le permet SHIELD sera meilleur pour des profondeurs supérieures. Le changement a lieu autour d'une profondeur 8.

2. Dans le cas où du *batching* serait possible ou souhaité, FV est généralement meilleur que SHIELD. Le batching consiste à évaluer un même circuit sur plusieurs données en même temps, grâce à un empaquetage de plusieurs clairs dans un chiffré unique.

Ce travail a comporté de nombreux challenges, se tenir à l'état de l'art en est un. Il est également difficile de comparer des schémas dont les structures ne comportent que peu de points communs. Notre travail est un des premiers du genre et appelle à être prolongé par une comparaison des performances réelles des schémas. Plusieurs sont déjà implémentés au sein de bibliothèques de fonctions open-source (Helib, SEAL, ...). Nous avons également initié l'implémentation de SHIELD en utilisant NFLlib pour permettre une comparaison la plus simple possible dans un travail futur.

Construction d'un nouveau schéma

La dernière contribution principale de cette thèse s'attèle à être force de proposition dans la dynamique communautaire de cryptographie homomorphe. Après avoir éprouvé la sécurité des schémas existants et comparé deux des propositions les plus prometteuses, nous avons travaillé à la conception et implémentation d'un schéma de dernière génération. Cette contribution est le fruit de mon séjour de recherche au sein du CWI d'Amsterdam. Dans cette collaboration avec Léo Ducas et Max Fillinger, j'ai assuré la partie design et réalisation de l'implémentation de ce schéma.

Un schéma de quatrième génération

Issue des travaux initiés par FHEW [DM15], la dernière génération des schémas complètement homomorphes apporte un profond changement de paradigme. Avant les dimensions du circuit à évaluer jouait un rôle prédominant dans le choix des paramètres des schémas. En effet le « bruit » intrinsèque aux chiffrés augmentant au cours de l'évaluation, il est nécessaire de choisir les paramètres pour que ce bruit n'empêche pas un déchiffrement correct. Or les schémas de quatrième génération ont pour caractéristique principale de maintenir un niveau de bruit constant au cours de l'évaluation du circuit. Cette dernière génération est encore très jeune et de nombreuses améliorations théoriques sont possibles. Notre travail est une extension de FHEW, orthogonal à celle proposée par [CGGI16, CGGI17].

L'évolution principale introduite dans ce schéma est l'utilisation de deux accumulateurs homomorphes dans deux anneaux de dimension modérée. Puis, grâce un produit tensoriel avant l'extraction de fonction, il est possible de rétablir une grande expressivité, tout en maintenant des coûts globaux très modestes. L'implémentation que nous discutons ci-dessous permet de réaliser des évaluations de portes logiques binaires sur 6 bits d'entrée en 6 secondes. La structure du schéma est présenté dans la figure 1.

De nombreux challenges pour l'implémentation

La figure 1 illustre bien l'hétérogénéité des objets à manipuler pour réaliser la porte dans sa globalité. Mes trois mois passés au CWI ont été dédié à réaliser une meilleure implémentation possible du schéma afin qu'il soit facilement évolutif (donc compréhensible par une personne extérieure) mais aussi performant pour révéler tout le potentiel des idées théoriques.

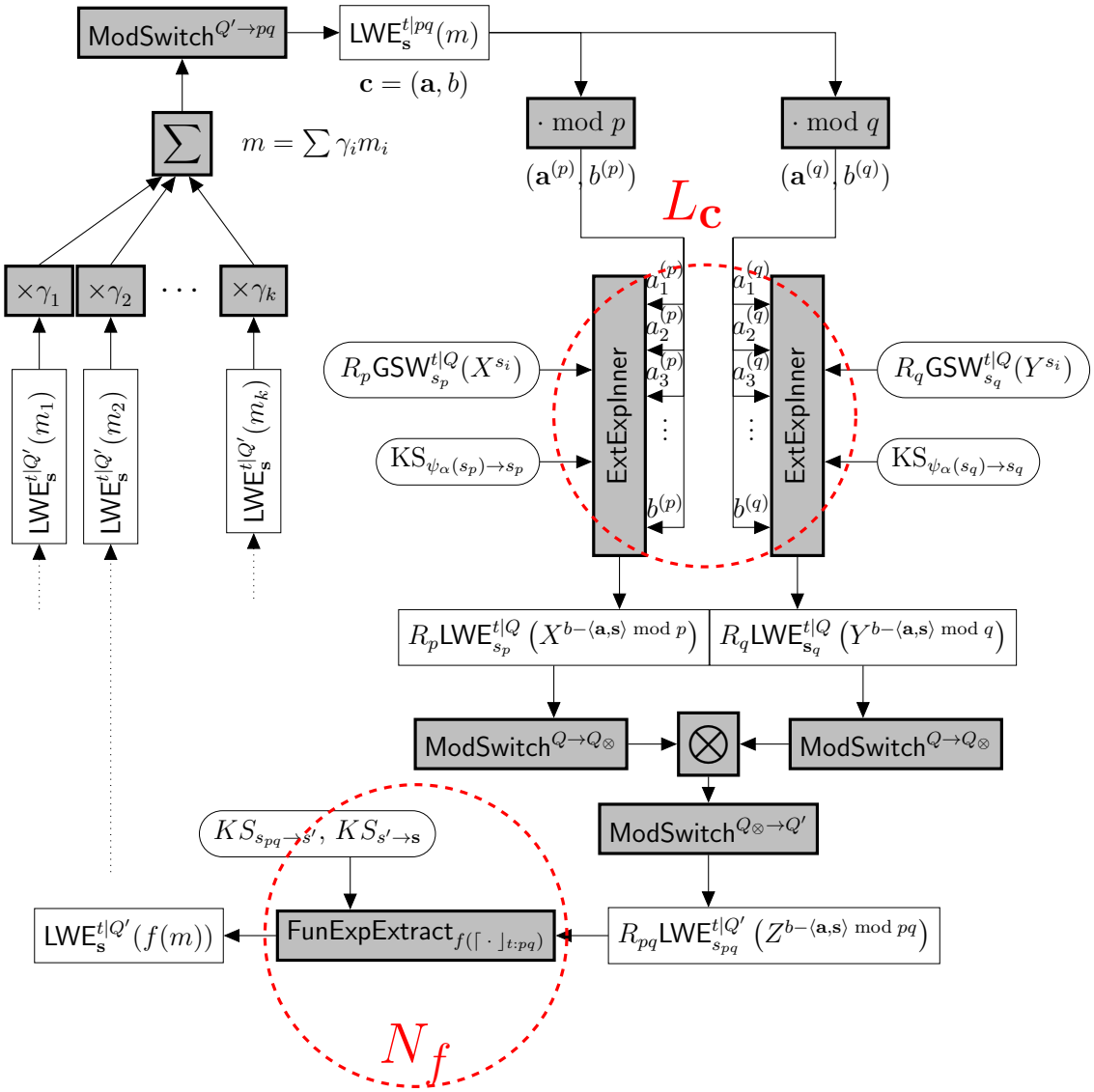
La réalisation a donc été faite en C++11 et l'usage de *templates* est la principale technique utilisée pour répondre aux objectifs de modularité et de performances. Des routines de bas-niveau ont été implémentés pour l'arithmétique modulaire et la manipulation d'éléments d'anneaux omniprésentes. Ensuite les objets cryptographiques en font usage.

Seules les transformés de Fourier, nécessaires aux multiplications d'éléments d'anneaux, sont réalisés par une bibliothèque externe, FFTW. Celle-ci nous a d'ailleurs apporté une contrainte de stabilité numérique. Étant donné que les transformés ont lieu sur des nombres à virgule flottante en double précision (i.e. 53 bits de mantisse), nous devons nous assurer que nos calculs ne dépassent jamais ce seuil. Le risque serait de faire apparaître une erreur supplémentaire dans le cryptosystème. Comme nos valeurs de modules pour l'arithmétique modulaire peuvent atteindre de très grandes valeurs (jusqu'à 2^{56}), nous devons dans ces cas scinder les éléments d'anneaux en leur partie de poids fort et celle de poids faible, procéder aux transformés sur chaque moitié et réaliser des multiplications terme à terme correctement (à la Karatsuba).

Performances

Notre première implémentation fonctionne pour 6 bits d'entrée avec les performances suivantes :

- Temps de génération de clés (une fois par programme) : 38 secondes ;



Sur la partie gauche, on voit les k bits d'entrée $m_1, \cdots, m_k$ qui sont combinés pour former $m \in \mathbb{Z}_t$. Sur le côté droit, on voit les deux Accumulateur Homomorphe ExtExpInner, qui réalise la partie linéaire $L_{\mathbf{c}}$ du calcul bootstrappé en mode CRT. Le résultat du tenseur alimente la partie non-linéaire du calcul $N_f : x \mapsto f(\lfloor tx/q \rceil \bmod t)$, i.e. FunExpExtract, où f est la fonction évaluée homomorphiquement. La sortie peut être directement fournie en entrée d'une porte suivante.
Les boites grises représentent les opérations, les boites blanches carrées les chiffrés, et celle arrondies les diverses clés.

FIGURE 1 – Aperçu du schéma.

- Temps d'évaluation d'une porte : 6.4 secondes ;
- Consommation mémoire : 9.2 Gio.

Le temps de la porte se décompose en 0.60 s pour chaque accumulateur (qui pourraient être calculés en parallèle), 4 s pour un changement de clés dans l'extraction de fonction, et 0.55 s pour les opérations liés au bit de sortie. Il serait donc possible de réaliser une fonction binaire de 6 bits vers 6 bits en une dizaine de secondes.

Perspectives

Quelques astuces d'implémentations pourraient encore être réalisées pour améliorer quelque peu les performances. En revanche, les techniques de RNS, utilisées récemment sur FV, pourraient quant à elles améliorer grandement les performances de notre schéma. Comparativement à TFHE [CGGI17], notre solution n'est en général pas la meilleure. Toutefois sur certains types de portes comme les portes à seuil, notre approche semble meilleure.

Chapter 1

Introduction

Computer science is no more about computers than astronomy is about telescopes

Edsger W. Dijkstra

1.1 The Chair of Naval Cyber Defence

This thesis work is an integral part of the overall research effort conducted within the Chair of Naval Cyber Defence[1]. This industrial chair has been launched in 2014 as a partnership between academic institutions and industrial players of the defense sector. Thales Group and Naval Group (formerly DCNS) are joined by the Institut Mines-Télécom Atlantique (formerly Télécom Bretagne) and the French Naval Research Institute and together they cover all scientific, academic and applications aspects. The motivation for this chair is to have a scientific approach to bring new solutions for improving cybersecurity on military ships.

Cyber-security is usually divided into cyber-protection, cyber-resilience and cyber-defense. The chair consists in several doctoral and post-doctoral students whose works cover this whole spectrum. The present work is about cyber-protection and the target application

[1] in French: `https://www.ecole-navale.fr/Chaire-de-cyberdefense-des.html`

of this thesis is to secure data aboard ships, military ships as a first motive but commercial ships as well.

1.2 A walk through the history of cryptology

1.2.1 What can crypto do for you?

Cryptology, from the Greek $\kappa\rho\upsilon\pi\tau o\varsigma$ (secret) and $\lambda o\gamma\iota\alpha$ (science) is the study of means to protect information (to use only broad terms). It is the union of cryptography (the creative, constructive part) and cryptanalysis (the assessment, attack counter-part). The first example of cryptography in history is the Caesar cipher, used by Emperor Julius Caesar during his conquests, more than two thousands years ago. The cryptanalytic technique that breaks this cipher is called *frequency analysis* and dates back from the 9th Century with Al-Kindi's work. Since these times, cryptologic work has been a cat-and-mouse game where cryptographers build more and more clever ways to protect information that cryptanalysts aim at breaking.

The overall protection objective is usually partitioned into three sub-objectives whose acronym C.I.A. is easy to remember. C stands for confidentiality: we do not want an eavesdropper to get the content of a protected exchange, I for integrity: the communication should not be altered by errors or attacks, and A for availability, legitimate entities should be able to access the information. In cryptology we call *primitives* the elementary building blocks that together form a complete protection solution. We refer here loosely about *entities* or *eavesdropper*, we shall in this work use the term *user(s)* when talking about the legitimate participants and *attacker* for the malicious, adverse party whose aim is to break the protection to some extent.

From an application viewpoint, we wish to limit the extra costs induced by the cryptographic protection to the minimum. As we shall see in this work, this is not always a trivial objective.

1.2.2 From paper-and-pencil to quantum... and post-quantum!

Let us now quickly review the last two thousands years of outstanding advances in cryptology. The paper-and-pencil area spans from Caesar cipher to the German AFDX cipher of the First World War. Depending on the technique, we speak about *code* or *cipher*. The former is about replacing meaningful word by others that only the legitimate recipient can understand, while the latter is about modifying the message content into something scrambled using *substitution* or *mixing*. A famous example of a code is Verlaine's poem *Les sanglots longs des violons de l'automne* that was broadcast by the BBC to French resistant forces during the Second World War. When the next verse was added *Blessent mon cœur d'une langleur monotone*, the message yields: "This is D-Day!". The Caesar cipher is an example of substitution, each letter is replaced by the one three position after in the alphabet: A becomes D, B becomes E and so on. So "This is D-Day" would be ciphered as *WKLV LV G-GDB*. Another example of cipher, using mixing only, say swap two consecutive letters in the message, would change our message into "HTSI SI D-DYA".

Of course none of these toy techniques are secure by today's standards, yet they have been state-of-the-art in their times. The Caesar cipher is called a mono-alphabetical substitution because encryption is about taking a letter replacement from one unique alphabet (the regular alphabet shifted by three in our example above). Another famous cipher of that kind is called Polybe Square which is about putting all the letters into a square and replacing them by the pair (row number, column number). These techniques were only broken by Al-Kindi's work on frequency analysis in the 9th Century when he realised that the most frequent letter in the original text remains the most frequent letter (or number) in the ciphertext. Thus, if you know the original text language you can easily decrypt the message.
After mono-alphabetical substitution, Vigenère introduced the poly-alphabetical cipher: the Vigenère cipher. This time, the letters of the message are not shifted by the same number of positions, but by a changing number of positions. For example the first would be shifted by eight positions, the second by twelve positions, the third by five, and the next by eight again and you loop like this for the whole message. This way, two identical letters in the ciphertext are not necessary the same in the cleartext. Analysing frequencies becomes more difficult. Now the attacker shall consider statistics on one every n letters in the ciphertext. Indeed in our example, every third letter is shifted by the same number of positions (eight, twelve or five). The first challenge is to find the number of shifts (i.e. the length of the key) before doing frequency analysis. The extreme version of this poly-alphabetical cipher, when each letter is shifted by an unpredictable number of positions (no loops and randomness), is called Vernam Cipher and is the only perfectly secure cipher ever to be built. Indeed this is perfectly secure as demonstrated by Shannon in his seminal work on information theory. The drawback of this cipher is that the key (the shifts to apply) is as long as the message, doubling the cost or time of the communications.
Another famous cryptographic proposal from the pencil-and-paper time is the AFDX, later ADFVX cipher. It is similar to the Polybe Square in that you write the letters in a 5×5 or 6×6 square whose columns and rows do not bear numbers but letters (A, F, D, X and V). To encrypt you apply this substitution, followed by a mixing operation on the resulting letters. This cipher was famously used in the First World War by the Germans and broken by the French George Painvain, only with pencil and paper.
The last famous cryptosystem to mention from the pre-computer area is the Enigma machine, again an extraordinary invention from the Germans that gave many troubles to the Allies. This mechanical device used rotors and wiring to apply a poly-alphabetical substitution to the messages and was not *broken* in the cryptographic sense. Indeed what was successfully used to recover messages was the brute-force technique that attempts every possible key against ciphertexts whose clear content was partially known. This brute-force attack was made possible by the Bombe of Alan Turing.

Later on, the development of communication devices and computers increased the need for information protection and provided new capabilities to protect... and to attack. The major problem of the techniques above is that they do not scale easily to world-wide communication. In all of them, the users have to share a piece of secret information (shifts, rotor configurations...), that we generally call the *secret key*. This means that, prior to any communication, there must have been a *key exchange* between the users. For military or highly critical communications, it is worth paying someone to physically deliver some key materials

(think about a carrier handcuffed to a locked suitcase for example) to support some amount of communication. But that cannot apply to private companies or individuals that would want to engage in secure communications. The solution to this was brought by Diffie and Hellman in 1976 (it was also invented by Cocks, from the British Intelligence Service, but the information was declassified only in 1997). The key exchange protocol allows two users to establish a common secret, using only public information exchanges. The first cryptosystem by Rivest, Shamir and Adleman (RSA) in 1978 provides another solution: it uses one *public* key to encrypt and another *private* to decrypt. Like this, everybody can encrypt a message to someone, who will be the only one able to retrieve the content. This is called *asymmetric cryptography* as opposed to *symmetric* cryptography. This marks the beginning of *modern cryptography*.

Today, cryptography provides many different primitives: symmetric encryption, asymmetric encryption, signature, key exchange, zero knowledge, random number generation... There are several standardised algorithms for these building blocks and, following the principles from Kerckhoffs, their design has been publicly reviewed and assessed by cryptographic experts for some time before being widely deployed. These proposals provide security proofs. Some are information theoretical results, some are hardness reductions to *problems* known or proven to be hard. The cryptosystem RSA, for instance, is believed secure because it somehow relates to the problem of factoring big integers, which has been a target from number theorists for centuries. Hence, it is assumed that no one will come up with an unexpectedly efficient algorithm to solve it. Thus, breaking RSA should be difficult. Hence most cryptographic components today rely on few of such hard problems to ensure security. The problems are: integer factorisation, discrete logarithm in cyclic groups. The only 100% proven secure algorithm is the Vernam cipher, also referred to as the *One-time pad*.
However, the ground-breaking result from Shor in 1994 came shaking the status-quo. His contribution known as Shor's algorithm is an efficient algorithm that works against both the discrete logarithm and the factorisation problem... but requires a quantum computer to work. Therefore we stand today with the knowledge that all deployed cryptographic components could be broken the day when a quantum computer is up and running Shor's algorithm. This is why, we see the emergence of *post-quantum* cryptography, meaning resistant to quantum attackers. It should not be mixed up with *quantum cryptography* which uses quantum properties to achieve security, for example quantum entanglement can be used to perform perfectly secure key exchange. This post-quantum cryptography has now been under study for several years, with first steps in 1978 with McEliece or 1996 with Ajtai. The U.S. National Institute of Standards and Technology (NIST) has annonced in 2015 a call for proposals of quantum resistant primitives, like it did in the past for other primitives, mostly with success.
Cryptographers turn to different mathematical objects in order to achieve quantum resistance, *lattices* are among them. In addition, lattices allow to build more advanced primitives such as *homomorphic encryption*. This kind of cryptography had been conjectured in 1978 but only realised in its full version, with lattices, in 2009 by Gentry. Homomorphic encryption is about encrypting data in such way that, once encrypted, they can still be operated upon. And when you decrypt the result of the computation, it yields the same result as if the computation had been done on the un-encrypted data. It was originally called privacy

homomorphism because it allows users to have their data being used by a third-parties (e.g. service provider) while keeping their data private. In a context of medical sensors or external storage and services (aka cloud), this cryptographic feature brings timely solutions.

1.3 This thesis

1.3.1 Outline

Fully Homomorphic Encryption (FHE) is therefore today a trendy topic, with high expectancies put into it, attracting much attention in the cryptology community. The aims of the present work is to participate in the effort to make FHE practical and study the added value such cryptography could bring to secure information aboard military ships.
Because of the pace of FHE improvements, the overall context has been moving quite a lot between the beginning and the end of this work. Nevertheless, we have conducted our study to the end. The approach we had was the following. First, we reviewed the most promising proposals, aiming first at understanding their security. Indeed the foundations of lattice-based cryptography are quite young, in cryptographic scale. Several years are usually needed for hardness assumptions to be thoroughly assessed. We contributed to this cryptanalytic effort, with a focus on lattice instances below FHE and lattice reduction. Next, we maintained a technological watch and performed a comparison of the best candidates as of end of 2016, with complete analysis of the costs and settings of each candidate, with real use cases in mind. Finally, we had the opportunity to work on the latest type of FHE scheme and developed a new FHE scheme, improving upon the state-of-the-art and provided a proof-of-concept implementation.

1.3.2 Roadmap

This manuscript follows to some extent the chronological order of the work of these last years.

- First in Chapter 2, we shall set the definitions and notation for the lattice theory we use in this work, together with a formal description of homomorphic encryption.
- Then we will start with our cryptanalysis efforts. Reviewing the state-of-the-art of lattice attacks, we shall present our special-purpose attack in Chapter 3. This work led to a publication in IndoCrypt 2017 [BF17]. Then in Chapter 4, as key component, we shall focus on lattice reduction, whose performance is directly linked to the attacker's advantage, and to which we contributed especially with involvements in the development of an open-source library `fplll`, which is included in Sage.
- Next, we move to our experience on choosing secure parameters for FHE in Chapter 5. We conducted a comparative study of the most promising schemes. Aiming at families of use-cases, we derived the concrete parameters of the schemes. We could then compare them depending on the scenario and provide guidance to the community of implementers on how to correctly setup the cryptography in their designs. This work has

been published in a special issue of *IEEE Transations on computers* on *Cryptographic Engineering in a Post-Quantum World* [**?**].

- Finally, improving upon the state-of-the-art, we developed a new scheme, more efficient, together with an open-source implementation, see Chapter 6. It was accepted to Africacrypt 2018 [**?**].

Chapter 2

Preliminaries

Let your memory be your travel bag.

2.1 Notation

In the thesis, we use the common mathematical sets, $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$, respectively the natural integers, the integers and the reals. For a real a, we note $|a|$ the absolute value of a and also $\lfloor a \rfloor$, $\lceil a \rceil$ and $\lfloor a \rceil$ respectively the rounding of a to the nearest integer smaller than a, bigger than a, and closest to a. For a positive integer $q > 0$ we denote $\mathbb{Z}_q$ the set of integers $(-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor]$.

Our log function is the logarithm in base 2.

For $n > 0$ integer, the matrix $\mathbf{I_n}$ refers to the identity matrix of size n. Capital bold letters are used for matrices, e.g. $\mathbf{B}$, and small ones for vectors, e.g. $\mathbf{v}$. We similarly write $\mathbf{B}$ for a matrix or for the ordered family of (row) vectors $\mathbf{B} = (\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n})$ using bold subscripts. For a vector $\mathbf{v}$, we refer to its components with italic letters v_i. We use $\langle \cdot, \cdot \rangle$ to speak about the inner product of two vectors of the same size.

For a vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n$, we define the Euclidean norm and note $\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$, and the infinity norm $\|\mathbf{v}\|_\infty = \max_{1 \leq i \leq n} |v_i|$. We may omit the subscript when it is clear from the context.

Rings

When dealing with a polynomial $\mathbf{p}$ of degree $n-1$, we use the *coefficient embedding* (unless specified) and assimilate it as a coordinate vector: $\mathbf{p} = (p_0, p_2, \ldots, p_{n-1})$. We will work with polynomials in a polynomial ring $R = \mathbb{Z}[x]/(f(x))$ for some $f \in \mathbb{Z}[x]$. We denote R_q the set of polynomials in R with coefficients in $\mathbb{Z}_q$.

For $\mathbf{a} \in R$ (or R_q) we define two norms:

- The *Coefficient norm* as $\|\mathbf{a}\| = \|(a_0, \ldots, a_{n-1})\| = \sqrt{\sum_{i=0}^{n-1} a_i^2}$.
- The *Operator norm* as $|\mathbf{a}| = \max_{b \in R \setminus \{0\}} \|ab\|/\|b\|$. We expand this notion to vectors $\mathbf{x} \in R^m$ with $m > 1$ by maximizing $\mathbf{y}$ over $R^m \setminus \{0\}$ and replacing the multiplication with the inner product over R.

Random variables

For a ring R and a polynomial $\mathbf{a}$, we say $\mathbf{a} \leftarrow U_R$ when $\mathbf{a}$ is sampled uniformly in R, $A \leftarrow B_R$ when it is sampled with binary coefficients and $\mathbf{a} \leftarrow D_{R,\sigma}$ when its coefficients are sampled independently from a (centred) discrete Gaussian distribution with width parameter σ, *i.e.* proportional to $\exp(-\pi x^2/\sigma^2)$.

We say that a real random variable X is *subgaussian with parameter* δ (or δ-*subgaussian*) if $\mathbb{E}[X] = 0$, and for all t, $\mathbb{E}[\exp(tX)] \leq \exp\left(t^2\delta^2/2\right)$. We consider only centred subgaussian variables ($\mathbb{E}[X] = 0$). Subgaussian random variables have the following well known properties Let X_1 and X_2 be subgaussian random variables with parameters δ_1 and δ_2, respectively.

- $X_1 + X_2$ is $(\delta_1 + \delta_2)$-subgaussian.
- If X_1 and X_2 are independent, $X_1 + X_2$ is $\sqrt{\delta_1^2 + \delta_2^2}$-subgaussian.
- aX_1 is $(|a|\delta_1)$-subgaussian.
- Subgaussian tail estimate: $P(|X_1| \geq \sqrt{2\lambda}\delta_1) \leq 2\exp(-\lambda)$.

Gadgets

Throughout this exposition we use a binary *decomposition* operation on ring elements, and the reverse. For simplicity we adopt the notation of gadget vector and matrix.

The *gadget vector* $\mathbf{g}^T$ of size K is set to $\begin{pmatrix}1 & 2 & 2^2 & \cdots & 2^{K-1}\end{pmatrix} \in R_d^K$. Reciprocally, we define $\mathbf{g}^{-T}$ as a function such that, for $\mathbf{w} \in R_d^n$, $\mathbf{V} = \mathbf{g}^{-T}(\mathbf{w})$ is a $(K \times n)$-matrix whose entries are ring elements with coefficients in $\{0,1\}$ such that $\mathbf{g}^T\mathbf{V} = \mathbf{w}$.

For some integer $n \geq 1$, the *gadget matrix* $\mathbf{G}_n$ is defined by $\mathbf{G}_n = \mathbf{I}_{n+1} \otimes \mathbf{g} \in R_d^{(n+1)K \times (n+1)}$.

$$\mathbf{G}_n^T = \begin{pmatrix} 1 & 2 & \cdots & 2^{K-1} & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 2 & \cdots & 2^{K-1} & \cdots & 0 & 0 & \cdots & 0 \\ & & \vdots & & & & \vdots & & \vdots & & & \vdots & \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 & 2 & \cdots & 2^{K-1} \end{pmatrix}$$

We define $\mathbf{G}_n^{-1}$ similarly to $\mathbf{g}^{-T}$: for $\mathbf{a} \in R_d^{n+1}$, we let $\mathbf{d} = \mathbf{G}_n^{-1}(\mathbf{a}) \in R_d^{(n+1)K}$ be the vector whose entries have coefficients in $\{0,1\}$ such that $\mathbf{d}^T \cdot \mathbf{G} = \mathbf{a}$. For convenience we write $\mathbf{G}_n = \mathbf{G}$ as n is typically clear from context.

2.2 Some lattice theory

In this section we synthesise the notions about lattices that will be used throughout this thesis.

2.2.1 General definitions

In general, a lattice $\mathcal{L}$ of dimension n is a discrete additive subgroup of $\mathbb{R}^n$. *Integer* lattices are discrete additive subgroups of $\mathbb{Z}^n$. In this work we only deal with the integer lattices and simply call them lattices.

Lattices (of size n) are usually represented by a basis $\mathbf{B}$, a set of n independent integer vectors $(\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n})$ of size n whose integer linear combinations generate the lattice.

$$\mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} v_i \mathbf{b_i} : v_i \in \mathbb{Z} \right\} = \left\{ \mathbf{B}^T\mathbf{v} : \mathbf{v} \in \mathbb{Z}^n \right\} = \mathbf{B}\mathbb{Z}^n$$

In our lattices, $\mathbf{B}$ is always a square integer matrix, $\mathbf{B} \in \mathbb{Z}^{n \times n}$. For most of the discussion we restrict to this full rank definition and will make it explicit when working with greater generating families of $m > n$ vectors.

For a lattice $\mathcal{L}$, we define its volume $\mathrm{Vol}(\mathcal{L})$ as the volume of the fundamental parallelepiped described by the basis vectors, $\mathcal{P}_{1/2}(\mathcal{L}) = \{\sum_{i=1}^n x_i \mathbf{b_i}, |x_i| < 1/2\}$. This volume is simply the absolute value of the determinant of a basis and is therefore a lattice invariant. We write

$$\mathrm{Vol}(\mathcal{L}) = \det(\mathcal{L}) = |\det(\mathbf{B})|$$

We often use projections onto subsets of basis vectors, so we note $\pi_{\mathbf{B},i}(\mathbf{v})$, the orthogonal projection of the vector $\mathbf{v}$ onto $\{\mathbf{b_1}, \ldots, \mathbf{b_{i-1}}\}$. Hence $\pi_{\mathbf{B},1}$ is the identity.

Given a lattice $\mathcal{L}$ we define its dual $\mathcal{L}^\star$ as the set of vectors $\mathbf{y} \in \mathbb{R}^n$ such that $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}$ for all $\mathbf{x} \in \mathcal{L}$. It holds that $\det(\mathcal{L}^\star) = 1/\det(\mathcal{L})$.

q-ary lattices. When studying cryptographic lattice problems, the lattices we mostly work with are called *q-ary*, because they are defined *modulo* some integer q (not necessarily prime). These lattices are defined as follows:

$$\mathcal{L}_q(\mathbf{B}) = \left\{ \sum_{i=1}^{n} v_i \mathbf{b_i} \bmod q : v_i \in \mathbb{Z} \right\} = \left\{ \mathbf{B}^T \mathbf{v} \bmod q : \mathbf{v} \in \mathbb{Z}^n \right\}$$

Hence, since we are working modulo q, we can equivalently consider that all the components of $\mathbf{B}$ and $\mathbf{v}$ are in $\mathbb{Z}_q$.

2.2.2 Gram-Schmidt Orthogonalisation (GSO).

We use several times the Gram-Schmidt Orthogonalisation of a basis. This algorithm takes the matrix to orthogonalise $\mathbf{B}$ and outputs the resulting matrix $\mathbf{B}^\star$ and a matrix μ (lower triangular with 1 on the diagonal) such that: $\mathbf{B} = \mu \times \mathbf{B}^\star$. It constructs $\mathbf{B}^\star$ so that its vectors verify:

- $\mathbf{b_1^\star} = \mathbf{b_1}$
- $\mathbf{b_i^\star}$ is the projection of $\mathbf{b_i}$ orthogonally to the subspace generated by the $i-1$ first vectors of $\mathbf{B}$. Formally $\mathbf{b_i^\star} = \mathbf{b_i} - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b_j^\star}$ where $\mu_{i,j} = \langle \mathbf{b_i}, \mathbf{b_j^\star} \rangle / \langle \mathbf{b_j^\star}, \mathbf{b_j^\star} \rangle$.

We have $\det(\mathcal{L}) = \prod_{i=1}^{n} \|\mathbf{b_i^\star}\|_2$. The algorithm works in polynomial time in the size of the matrix.

2.2.3 Fundamental results

For any lattice, we define *Minkowski's successive minima* $\lambda_i(\mathcal{L})$. They are the radiuses of the smallest centred in zero balls containing i linearly independent vectors, i.e. $\lambda_1(\mathcal{L})$ is the norm of a shortest non-zero vector of $\mathcal{L}$, $\lambda_2(\mathcal{L})$ the norm of the second shortest vector and so on.

The *Gaussian Heuristic (GH)* states that the volume of the intersection of a set S and a lattice $\mathcal{L}$ is: $\mathrm{Vol}(S \cap \mathcal{L}) \approx \mathrm{Vol}(S)/\mathrm{Vol}(\mathcal{L})$. Especially, it yields a useful estimate for the first minimum:

$$\lambda_1(\mathcal{L}) = \sqrt{\frac{n}{2\pi e}} \mathrm{Vol}(\mathcal{L})^{1/n}$$

Finally, we introduce the *Root Hermite Factor* of a basis that we note δ_0. For a given basis $(\mathbf{b_1}, \ldots, \mathbf{b_n})$, it is defined such that:

$$\|\mathbf{b_1}\| = \delta_0^n \cdot \mathrm{Vol}(\mathcal{L})^{1/n}$$

It is clearly dependent of basis used to represent the lattice and, as we shall see next, plays a central role when speaking of basis *quality*.

Hermite's work from the 19th century provides us with the following inequality:

$$\frac{\lambda_1(\mathcal{L})}{\mathrm{Vol}(\mathcal{L})^{1/n}} \leq \left(\frac{4}{3}\right)^{\frac{n-1}{4}}$$

Therefore we define the *Hermite constant* γ_n as the extremum of the left side hand ratio over all lattices of a given dimension n. It is folklore that γ_n grows linearly in n, yet exact value for it are only known for n from 1 to 8, and 24.

2.3 Lattice hard problems

The study of lattices by mathematicians goes far back in time with sphere packing analysis, but it is only recently that cryptologists got interested into them. First, with Coppersmith [Cop96] who used lattices to break RSA under some assumptions. Then Ajtai [Ajt96] opened the way to lattice-based cryptography, showing the existence of NP-hard problems in lattices. We review now the landscape of these hard problems, that support today all primitives of lattice-based cryptography, homomorphic encryption among them.

2.3.1 Shortest Vector Problem (SVP) and consorts

Shortest vector problems

The first (and easiest to state) of these problems is called the *Shortest Vector Problem (SVP)*. This problem asks to find a shortest non-zero vector $\mathbf{v}$ in a lattice $\mathcal{L}$, given a basis $\mathbf{B}$. It shall satisfy

$$\|\mathbf{v}\| = \lambda_1(\mathcal{L})$$

Note that due to the group structure of lattices, *the* shortest vector is not unique, never. In every lattice, if $\mathbf{v}$ realises the first minimum, then $-\mathbf{v}$ does it too. There are also lattices were several linearly independent vectors have all the shortest norm (e.g. the hexagonal lattice).

The result of Ajtai [Ajt98] is that the *exact* version of SVP is NP-hard.

Approx-SVP. If we relax the condition to finding a vector $\mathbf{v}$, such that $\|\mathbf{v}\| \leq \gamma\lambda_1(\mathcal{L})$ for some $\gamma > 1$, we have the *Approximate Shortest Vector Problem (Approx-SVP)*. The greater the constant γ, the easier the problem. Assuming the Gaussian Heuristic, we can reformulate

$$\begin{aligned}\|\mathbf{v}\| &\leq \gamma \cdot \lambda_1(\mathcal{L}) \\ &\leq \gamma \cdot \sqrt{\frac{n}{2\pi e}}\mathrm{Vol}(\mathcal{L})^{1/n}\end{aligned}$$

Hence, a lattice reduction that achieves a root-Hermite factor of $\delta_0^n = \gamma\sqrt{\frac{n}{2\pi e}}$ solves Approx-SVP with approximation factor γ. Since LLL achieves $\delta_0^n = (4/3)^{(n-1)/4}$ in polynomial time, we can say that Approx-SVP with exponential approximation factor is not hard. There are several results about Approx-SVP hardness for intermediate approximation factor [Mic01, AR05].

Hermite SVP. For Approx-SVP, the factor bounds the short vector size with respect to the first Minkowski minimum, that we do not always know in advance. Therefore, if we refer to the volume instead: $\|\mathbf{v}\| \leq \gamma \text{Vol}(\mathcal{L})$, we define the *Hermite Shortest Vector Problem* with approximation factor γ.

Unique SVP. As we shall see later, lattice-based primitives are such that the lattice contains one unexpectedly short vector. This might turn out to be a much easier problem that we refer to as the *Unique Shortest Vector Problem (uSVP)*, which is about finding a shortest vector when $\lambda_2(\mathcal{L}) > \gamma\lambda_1(\mathcal{L})$.

Gap SVP. Another variant of Approx-SVP is called GapSVP. This is a decisional version. It asks to determine, for a fixed factor γ, if a given lattice $\mathcal{L}$ verifies $\lambda_1(\mathcal{L}) \leq c$ or $\lambda_1(\mathcal{L}) \geq \gamma \cdot c$, with the promise that the lattices fall into one or the other case.

Closest vector problems

Where SVP is the homogeneous problem to find a lattice vector close to $\mathbf{0} \in \mathcal{L}$, we can also ask to find a lattice vector close to some target $\mathbf{t} \in \mathbb{R}^n$. More formally, given a basis $\mathbf{B}$ of a lattice $\mathcal{L}$ of dimension n and a target $\mathbf{t} \in \mathbb{R}^n$, find the vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{t} - \mathbf{v}\|$ is minimal.

Approx-CVP. As for SVP, we can define an *Approximate Closest Vector Problem* for an approximation factor γ. This version is about finding a vector $\mathbf{v} \in \mathcal{L}$ such that

$$\|\mathbf{t} - \mathbf{v}\| \leq \gamma \cdot \text{dist}(\mathcal{L}, \mathbf{t})$$

where $\text{dist}(\mathcal{L}, \mathbf{t})$ is the distance from $\mathbf{t}$ to *the* closest vector from $\mathbf{t}$ in lattice $\mathcal{L}$. Both exact and approximate CVP are hard within constant factor [ABSS93], furthermore any hardness result on CVP implies the same hardness for SVP [GMSS99], i.e. SVP is not harder than CVP.

Bounded Distance Decoding. If in addition, we have the promise that the closest vector is unique and that $\text{dist}(\mathcal{L}, \mathbf{t}) < \gamma\lambda_1(\mathcal{L})$, we speak about the *Bounded Distance Decoding (BDD)* problem. This problem is one of the closest to the instances we see later, and we have a hardness result for it [LM09].

2.3.2 Learning with errors (LWE)

Using the previous problems, it is possible to construct some primitives for lattice-based cryptography, yet cryptographers looked for more expressive problems that would enable more advanced primitives. In 2005, Regev introduced the *Learning with errors (LWE)* problem [Reg05], which is about solving noisy linear equations. More formally, given two positive integers n (dimension), q (modulus) and a probability distribution χ over $\mathbb{Z}$, we define the $\text{LWE}_{n,q,\chi}$ problems as follows:

For a fixed secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, we construct m samples $(\mathbf{a}, b) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, where $\mathbf{a}$ is sampled uniformly in $\mathbb{Z}_q^n$ and e is sampled from χ and considered in $\mathbb{Z}_q$. The *Search* problem is about recovering $\mathbf{s}$ from the available samples and the *Decision* problem is about distinguishing such samples from uniform samples in $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Several hardness results have been demonstrated: classical or quantum, for different sizes of modulus q, in an *normal form* (secret and error are sampled from the same distribution). The condition that always remains is that the error distribution shall not be to small, otherwise the problem is not hard. To the extreme, if we have $e = 0$, it is only a matter of linear algebra to recover $\mathbf{s}$.

A former problem: *Learning Parity with Noise (LPN)* [BKW03] is a particular case of LWE problem where $q = 2$ and χ is the Bernouilli distribution over $\{0, 1\}$.

Another convenient way to express the LWE problem with m samples is to adopt a matrix notation. With $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ where each row is an $\mathbf{a}$ from the previous definition and $\mathbf{b}$ and $\mathbf{e}$ are column vectors of the b's and e's, we can write $(\mathbf{A}, \mathbf{b}) = (\mathbf{A}, \mathbf{As} + \mathbf{e}) \in \mathbb{Z}_q^{m \times (n+1)}$. In LWE-based schemes, $\mathbf{A}$ relates to the lattice basis.

2.3.3 Learning with errors over rings (Ring-LWE)

The use of LWE to construct schemes often implies many samples and therefore the matrix $\mathbf{A}$ tends to be expensive to store or share. Therefore, the community tried to find ways to gain efficiency in this matter. A solution is to add structure to the lattice and consider, for instance, *ideal lattices*. In this setting, the description of the lattice requires only a vector $\mathbf{a}$ instead of a matrix $\mathbf{A}$ decreasing the costs of the scheme from $O(n^2)$ down to $O(n)$. More precisely, we define the *LWE problem over Rings (Ring-LWE)* [LPR10] as follows.

Given a ring R of degree n over $\mathbb{Z}$ and a positive integer q, we call $R_q = R/qR$ the quotient ring. We also fix χ a probability distribution over R. Then, for a fixed $s \in R_q$, we construct m samples $(a, b) = (a, a \cdot s + e \mod q)$ where a is sampled uniformly in R_q and e is sampled from χ. The search problem asks to recover s from the samples and the decision version is to distinguish such samples from uniform samples in R_q^2.

In terms of security, Ring-LWE enjoys security reductions to SVP in *ideal* lattices. So this cannot be harder than regular LWE. Yet as of today there is no significantly better algorithm to solve SVP in the ideal case. The algebraic structure is nevertheless an aspect that requires much attention. The original problem statement involves a fractional ideal $R^\vee$ dual to R and therefore the statement above is equivalent from an application perspective but differs to some extent when doing their rigorous analysis.

2.3.4 Miscellaneous

Module-LWE

Working in highly structured lattices, e.g. ideal lattices, or even more so with cyclotomic rings, brings many concerns about real hardness of these instances. Therefore we see several attempts to trade-off between the higher efficiency of the structure and their weaker hardness results. One is to shift to non-cyclotomic rings and the other to *module* lattices and define

module-LWE [LS15] for instance. This problem generalizes Ring-LWE where the elements are not living in R_q of dimension n but in R_q^d of dimension nd. This provides easy trade-offs and we see several primitive based upon it.

Middle-Product LWE

Another way to shift the LWE formalism is to change the binary operator between a and s. Instead of using the regular product, as in Ring-LWE for example, the idea is to use the *middle-product* operator. The middle product of size d of a and b, two polynomials of degree d_a and d_b is:

$$\mathrm{MP}(a,b) = \left\lfloor \frac{ab \mod x^{k+d}}{x^k} \right\rfloor, \text{ where } 2k = d_a + d_b - d - 1$$

Interesting hardness results has been demonstrated for this problem [RSSS17] together with efficient primitives.

Short Integer Solution

Another expressive problem, somehow dual to LWE, is the *Short Integer Solution (SIS)* problem. It was introduced even before LWE by Ajtai's work [Ajt96], but since it is more useful to construct signatures than encryption schemes, we prefer to mention it only as extra information, not at the core of this thesis work. Informally, SIS asks to find a linear combination, with small coefficients, of given random elements that yields zero.

More formally, as with LWE we have a dimension n, a modulus q and m samples $(\mathbf{a_i})_{1\leq i\leq m} \in (\mathbb{Z}_q^n)^m$, SIS asks to find $(\mathbf{s_i})_{1\leq i\leq m}$, each of norm $\|\mathbf{s_i}\|$ smaller than some threshold t, such that $\sum \mathbf{a_i}\cdot\mathbf{s_i} = \mathbf{0} \in \mathbb{Z}_q^n$.

SIS enjoy hardness reduction to GapSVP as LWE. It also has an inhomogeneous version where the combination target is no longer $\mathbf{0}$ but some vector in $\mathbb{Z}_q^n$, and also ring and module variants.

2.4 Presentation of FHE

We shall now introduce the core of this thesis: homomorphic encryption (HE). First with an overview on the history of the different scheme proposals that led to the current state-of-the-art in the subject. And then, we shall present the different common components of homomorphic encryption schemes, using that from Fan and Vercauteren [FV12] to illustrate.

2.4.1 Homomorphic Encryption history (2008-2017)

Homomorphic encryption dates back to the notion of privacy homomorphism whose existence was conjectured in the late 70s [RAD78]. The aim is to allow computation to take place over private data, e.g. encrypted data. The situation back then was that encryption scheme only allowed encryption and decryption. Any operation on ciphertext would give a completely uncontrolled decrypted plaintext, or worse, prevent its decryption.

The first step, to perform *any* operations, was to allow *one* type of operation. This is what we call *partially* homomorphic encryption whose most famous examples are the cryptosystem RSA [RSA78] and that of Paillier [Pai99]. Indeed, two RSA ciphertexts can be multiplied together and yields an encryption of the product of their respective plaintexts. We say that RSA is homomorphic for multiplication. Similarly Paillier is homomorphic for addition. Other schemes provide homomorphism for multiplication or addition, but it is only in 2008 with the work from Aguilar-Melchor et al. [AGH10] that it became possible to do both additions and multiplications on ciphertexts, at least a bounded number of them. It creates the concept of *Somewhat Homomorphic Encryption (SHE)* whose schemes permits a number of operations known in advance. Next, in 2009, Gentry [Gen09] set the first stone to *Fully Homomorphic Encryption (FHE)*, where no more constraints pertains to the computation. Thanks to the novel technique of *bootstrapping* which consists in homomorphically decrypting the ciphertext to refresh it, it becomes possible to evaluate any circuits on encrypted data. At least in theory because this early proposal is not efficient at all, but opened the road to many improvements. The first implementation of the ideas of Gentry is due to Smart and Vercauteren [**?**].

The first improvement was introduced with work from van Dijk et al. [vDGHV10], dubbed *FHE over the integers*. Instead of lattices like Gentry, they only use integers to construct their scheme, whose security relies on the hardness of *Approximate Greatest Common Divisor (AGCD)* problem. This problem asks, given several elements which are almost multiples of an integer p, to recover p. Several improvements have been brought to the original DGHV scheme until 2012 [CMNT11, CNT12].

Then, many homomorphic encryption scheme were designed over LWE and ported to the ring setting for efficiency. Brakerski and Vaikuntanathan's scheme introduced several novel technique such as the *modulus switching* [BV11b, BV11a]. This operation allows to scale down the noise inherent to the ciphertext which increases at each operation. This postpones the need for a bootstrapping. Also their scheme shows a linear increase of the error magnitude with respect to the *circuit depth*, unlike Gentry's scheme which has quadratic increase. Next, the modulus switching was pushed even further in BGV scheme [BGV12], where a complete ladder of moduli is used, so that after each multiplication, the ciphertext is scaled down to the next modulus. This is what we call a *leveled HE scheme*. BGV is, as of today, one of the most promising scheme and an implementation is available in the IBM Research library HElib [Hal] with many implementation tricks [GHS12a, GHS12b, SV14]. Next, leaving modulus switching aside, Brakerski presented a scale invariant scheme over LWE [Bra12], later transposed to Ring-LWE by Fan and Vercauteren [FV12]. FV is also very promising and is implemented by Microsoft Research in SEAL library [LCP17] and using NFLlib [Cry]. It also enjoys a fast variant using Residue Number System (RNS) [BEHZ16]. Both BGV and FV belongs to the second generation of homomorphic schemes, Gentry's was the first generation.

We have also seen homomorphic schemes inspired by NTRU encryption scheme [HPS98]. YASHE [BLLN13], which has similarities with FV, sounded very interesting, even at the beginning of this thesis. It was the original scheme in SEAL library. However, in 2016, the subfield/sublattice attack [ABD16, KF17] damaged completely the *overstretched* NTRU assumption that sustained YASHE.

Later, third generation schemes were introduced. They remove the need for relinearisation of the second generation proposals. GSW [GSW13] extends upon Brakerski's scheme [Bra12], and SHIELD [KGV16] is its ring variant. Similarly F-NTRU [DS16] extended YASHE but became obsolete with the subfield/sublattice attack. Hence, SHIELD is the only interesting third generation scheme. It involves matrices where second generation schemes needed only vectors, therefore the minimal cost is higher, but thanks to the absence of relinearisation, the cost increases far slower than for previous generation schemes.

All the proposals from above are SHE, meaning that given a choice for their parameters, they can evaluate circuits of bounded complexity. The bootstrapping technique of Gentry can then be applied if the scheme parameters allow at least to evaluate the decryption procedure. For example, say that the procedure requires 8 multiplications, the scheme parameters shall be so that at least 9 multiplications are possible. In such cases, we can transform the SHE scheme into a FHE scheme. Sometimes it is not necessary if the application has a fixed number of multiplications for example.

Yet, the latest line of work is about *bootstrapped* schemes [AP14, DM15, BR15, CGGI16]. They have the property to maintain the noise level in ciphertext across the evaluation, removing the need for operations dedicated to bootstrapping. It is somehow built into the scheme. This proposals appeared quite recently and we have not yet applications using them, but they should ultimately be the best homomorphic encryption schemes.

2.4.2 Dimensioning constraints

As with any encryption scheme, the context of application is central to correctly setup the scheme and choose its parameter(s). From the end-user perspective, the scheme should adapt to two elements: the programme to evaluate $\mathcal{P}$ and the security level λ. Security is commonly evaluated in bits, λ bits of security means an attacker will face a cost of 2^λ to break the scheme.

Concerning the programme, we (designers) classically view it as a circuit with elementary gates, which then can be viewed as a polynomial function on the input data. For SHE/SHE schemes (e.g. BGV, FV) the main feature to consider is the circuit *depth* L. We usually define it as the maximum number of multiplications that an input data goes through to produce the output. Indeed, the multiplications are the operations that have the most impact on the noise in ciphertexts. And since we want to keep this noise under control to maintain decryption correct, this number of multiplications is the most important feature. For bootstrapped schemes, which usually offer more expressive gates that just addition/multiplication, the concerns are a bit different. We shall come back to that later in this work.

For cases like FV, we now have a depth L and security parameter λ and shall derive scheme parameters so that the computation will be both correct and secure. We shall see later that these two goals bring opposite conditions that may turn out contradictory. In the end, since our schemes are based on (Ring-)LWE, the parameter choice will give us at least a dimension n, a modulus q and an error distribution χ.

2.4.3 One example: the Fan-Vercauteren scheme (FV)

To illustrate the different components of a homomorphic encryption scheme, we take the one of Fan and Vercauteren [FV12], which is central for this thesis work.

FV is the ring variant of the scheme from Brakerski [Bra12] and therefore all operations of FV are done in some ring R. The original description of FV sets R to be the polynomial ring $R = \mathbb{Z}[x]/(f(x))$ for some monic irreducible polynomial $f \in \mathbb{Z}[x]$ of degree n. For easy exposition, we often choose f to be cyclotomic: $f(x) = x^n + 1$ with n a power of 2. The elements at hands are polynomials of degree n, and we restrict to R_q where the polynomials have their coefficients in $\mathbb{Z}_q$, for some positive integer q.

Key generation

The secret key generation is a fairly simple operation, but for efficiency purposes, it may differ between implementations. The original procedure is to sample the coefficients from a discrete Gaussian distribution χ_s of standard deviation σ_s.

$$s_k = \mathbf{s} = (s_1, \cdots, s_n) \in R_q, \text{ where } \forall i, s_i \leftarrow \chi_s$$

Sometimes we see that $\mathbf{s}$ is binary (coefficients in {0, 1}) or ternary (coefficients in {-1, 0, 1}) or even sparse (only a fixed number of coefficients are non-zero).

The public key is then derived from s_k. Sample a uniformly random $\mathbf{a} \in R_q$ and an error $\mathbf{e}$ from a discrete Gaussian distribution χ_e of standard deviation σ_e. Then define

$$p_k = (\mathbf{p_0}, \mathbf{p_1}) = (-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) \mod q, \mathbf{a}) \in R_q^2$$

We notice that this public key is a Ring-LWE sample. The error distribution may or may not be the same as the secret. However, error sampling is not trivial and should be done with care, especially while taking into account the ring R. As a reminder, the hardness results on Ring-LWE considers not R but $R^\vee$ and this difference should be understood when implementing the error sampling.

Encryption and decryption

In FV, the plaintext space is R_t, the set of polynomials of degree n with coefficients in $\mathbb{Z}_t$ for some positive integer t, and the ciphertext space is R_q. We note $\Delta = \lfloor q/t \rfloor$.

To encrypt a message $\mathbf{m} \in R_t$, sample a binary polynomial $\mathbf{u} \in R_2$ and two errors $\mathbf{e_1}$ and $\mathbf{e_2}$ from χ_e, then set

$$\mathbf{c} = (\mathbf{c_0}, \mathbf{c_1}) = (\Delta \cdot \mathbf{m} + \mathbf{p_0} \cdot \mathbf{u} + \mathbf{e_1}, \mathbf{p_1} \cdot \mathbf{u} + \mathbf{e_2}) \in R_q^2$$

It is worth to note that homomorphic encryption schemes are often public encryption scheme and always random, two encryptions of the same plaintext must be independent from one another.

For decryption, compute

$$\begin{aligned} \mathbf{c_0} + \mathbf{c_1} \cdot \mathbf{s} &= \Delta \cdot \mathbf{m} + \mathbf{p_0} \cdot \mathbf{u} + \mathbf{e_1} + (\mathbf{p_1} \cdot \mathbf{u} + \mathbf{e_2}) \cdot \mathbf{s} \\ &= \Delta \cdot \mathbf{m} + (-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) + \mathbf{a} \cdot \mathbf{s}) \cdot \mathbf{u} + \mathbf{e_1} + \mathbf{s} \cdot \mathbf{e_2} \end{aligned}$$

which yields $\Delta \cdot \mathbf{m}$, plus small polynomials. Hence by scaling and rounding we can recover

$$\mathbf{m} = \lfloor (\mathbf{c_0} + \mathbf{c_1} \cdot \mathbf{s}) \times t/q \rceil \in R_t$$

We can see that for decryption to work, the errors shall remain *small*. We will come back more extensively on this aspect later in this work.

Homomorphic addition

Performing addition with FV is not a complex operation, as is often the case with other SHE. Addition is a linear operation, so if we have $\mathbf{c}^a$ and $\mathbf{c}^b$ two encryptions of $\mathbf{m}^a$ and $\mathbf{m}^b$ respectively, then it is easy to verify that

$$\mathbf{c}^+ = (\mathbf{c_0}^+, \mathbf{c_1}^+) = (\mathbf{c_0}^a + \mathbf{c_0}^b, \mathbf{c_1}^a + \mathbf{c_1}^b) \in R_q^2$$

is a ciphertext for $\mathbf{m}^a + \mathbf{m}^b$. The input noises are added to form the output noise.

Homomorphic multiplication

Multiplication is a more involved process. Indeed, we would like that the decryption of $\mathbf{c}^\times$ yields the same as the product $\mathbf{m}^a$ and $\mathbf{m}^b$. In other words we wish that

$$\begin{aligned} \lfloor (\mathbf{c_0}^\times + \mathbf{c_1}^\times \cdot \mathbf{s}) \times t/q \rceil &= \lfloor (\mathbf{c_0}^a + \mathbf{c_1}^a \cdot \mathbf{s}) \times t/q \times (\mathbf{c_0}^b + \mathbf{c_1}^b \cdot \mathbf{s}) \times t/q \rceil \\ &= \lfloor \mathbf{c_0}^a \cdot \mathbf{c_0}^b + (\mathbf{c_0}^a \cdot \mathbf{c_1}^b + \mathbf{c_1}^a \cdot \mathbf{c_0}^b) \cdot \mathbf{s} + (\mathbf{c_1}^a \cdot \mathbf{c_1}^b) \cdot \mathbf{s}^2 \times t^2/q^2 \rceil \end{aligned}$$

However we have in the right-hand side a term in $\mathbf{s}^2$ that makes this straightforward product a non-valid ciphertext. This is why we need to perform a *relinearisation* of the quadratic term in order to get $\mathbf{c_0}^\times$ and $\mathbf{c_1}^\times$ so that the equality holds. For other schemes this operation may be referred to as a *key switching* because the trick is about taking the part that is *somehow encrypted* under $\mathbf{s}^2$ and getting it back encrypted under $\mathbf{s}$.

Relinearisation. To do this we need to make another, non negligible, assumption for security: we need *circular security*, that is to say we need to encrypt elements highly related to $\mathbf{s}$ under that same key $\mathbf{s}$. This extra assumption is universal in homomorphic encryption scheme and it would be a huge leap forward to describe a new scheme that does not need it.

In practical terms, we can construct an almost encryption of $\mathbf{s}^2$ by $\mathbf{c}^r = (-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) + \mathbf{s}^2, \mathbf{a}) \in R_q^2$ for some $\mathbf{a}$ and $\mathbf{e}$ as before. It holds that $\mathbf{c_0}^r + \mathbf{c_1}^r \cdot \mathbf{s} = \mathbf{s}^2 + \mathbf{e}$. Provided this $\mathbf{c}^r$ is public, we can then replace the $\mathbf{s}^2$ terms by linear terms at the cost of an extra error. Yet, this is not good enough because this extra error will be multiplied by the term in front of $\mathbf{s}^2$ in the equation above, i.e. a (non-small) uniform element of R_q, so the error will blow up. The way to handle this is to slice this term (consider its decomposition in some basis ω), this process is referred to as the *gadget decomposition*. Then, with the appropriate relinearisation keys, it is still possible to get rid of the $\mathbf{s}^2$ while maintaining the error growth under control. As we shall see later, the choice of ω is a trade-off choice between a bigger relinearisation key to share and a smaller error growth.

Chapter 3

Recipe for a new attack

In this chapter we adopt an attacker posture in order to assess the security of homomorphic encryption schemes. In the previous chapter, we said that homomorphic encryption schemes sit on hardness results and are proven secure, even against quantum attackers. Such statements speak about asymptotic behaviour; they mean that it is possible to choose the parameters of a problem such that solving it is infeasible within some resource limit. In concrete terms it says that if we pick a parameter (e.g. a dimension) big or small enough, the attacker will not be able to solve the problem with his available resources.

Therefore, when given an concrete application, we have to put a cursor on the security level we want to achieve. We usually speak in terms of *bit security*, noted λ. Then, we say that our scheme has a security of λ bits of security, if we estimate the *cost* of the attack (e.g. time, space, money...) to be in the order of magnitude of 2^λ. As a result, we need to have some link between the parameters of the scheme and the security level. This link is necessarily obtained by a similar link between the parameters of the underlying lattice problems (SVP, LWE, ...) and their *concrete hardness*. In this work, we are especially interested in that of LWE and its ring variants.

The present chapter begins with a review of the known attacks against (Ring-)LWE. Then we present our first contribution: a dedicated attack on LWE instances as found in homomorphic encryption settings. This work was accepted to IndoCrypt 2017 conference [BF17]. Finally, we report some contributions made on lattice reduction, which is a central

part in most of the attacks. These contributions were mostly done on the `fplll` library [dt17] during coding days.

3.1 Review of the existing attacks

Just before this thesis began, Albrecht et al. released the survey *Concrete hardness of Learning with Errors* [APS15] which served as an excellent springboard in the subject for us. In addition to their work, they provided and maintain an `lwe-estimator` [Albb] which computes the expected attack costs against LWE with the input parameters. As we discussed in the previous chapter, Ring-LWE instances can be viewed as special cases of LWE instances, so all the attacks against LWE can be used against Ring-LWE.

3.1.1 Against LWE

Let us now review the different attacks against LWE.

Algebraic attack – Arora-Ge

The first attack we present here is that of Arora and Ge [AG11]. It is an algebraic attack that establishes a system of non-linear noise-free polynomials, whose roots are the secret $\mathbf{s}$. Solving the system allows to recover the secret. With Gröbner bases it is possible to optimise the resolution [ACF+15a].

This attack assumes that the errors remain in some small range $[-e; +e]$ because this value e determines the degree of the polynomials and also the number of samples we need for a valid resolution. Consequently, this is an interesting attack, because it is subexponential when the LWE error distribution is smaller than what the hardness reduction requires. It strengthens the fact that errors should not be too small. At least in theory, because this attack comes with large constant costs and despite its subexponential time, it is never an efficient way to attack LWE.

Combinatorial attacks – BKW

The Blum-Kalai-Wasserman method [BKW03] was introduced against the Learning Parity with Noise problem, a special case of LWE we mentioned earlier. From a global perspective, BKW-like algorithm are about using many LWE samples to create collisions and exhibit relations between samples that allow to reveal the secret. Since its introduction, many improvements have been brought to this line of work [AFFP14, ACF+15b, DTV15, GJS15, KF15]. The main drawback is that these algorithms require many samples to work.

Lattice attacks – Primal and dual

Last, but not least, the lattice attacks. This family regroups the methods working with the structure of lattice *behind* the LWE samples and attempting to solve the equivalent BDD, CVP or SVP instances. The most natural is that of *decoding*. Assuming we are given an

LWE sample (e.g. a public key from Fan-Vercauteren scheme) $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ and the description of the lattice $\mathbf{A}$, we try to remove the error $\mathbf{e}$ to recover the lattice point $\mathbf{A} \cdot \mathbf{s}$ which then allows to recover $\mathbf{s}$. This is solving a BDD instance. Below we present more extensively the algorithms [Bab86, LP11, GNR10, LN13] for this and our contribution.

As we shall see, in order to decode it is sometimes better to *embed* the lattice. That is, to consider the elements of $\mathbf{A}$ in another, bigger, lattice. This is what Kannan [Kan87], or Bai and Galbraith [BG14] do. We will present these techniques in more details below. It allows to reduce BDD to uSVP [AFG14, AGVW17].

Finally, instead of working in the *primal* or direct lattice, it is also interesting to work with the *dual* lattice, as in [Alb17].

3.1.2 Against Ring-LWE

As we said above, Ring-LWE can be considered as a special case of LWE, but the additional ring structure may make dedicated attacks possible. We recently saw this trend in the following works [EHL14, ELOS15, CLS15, CLS16, CIV16] which were summed up by Peikert [Pei16]. All these works take advantage of the error distribution shape in the ring. Hence, none of them invalidates the hardness result about Ring-LWE [LPR10] which specifies that the errors should be sampled in $R^\vee$ and not in R. Nevertheless, the difference allows to mount attacks against careless error sampling. Peikert shows how to instantiate Ring-LWE securely.

When presenting the ring-based homomorphic encryption schemes, we mentioned YASHE [BLLN13] and F-NTRU [DS16] that are based on the hardness of the NTRU assumption (not only on that of Ring-LWE). However, for performance considerations, the choices of parameters that the authors advertised were not compliant with the hardness result about NTRU [SS11]. Therefore we saw successful attacks against them [ABD16, KF17].

3.2 Mounting our own dedicated attack

We just described the landscape of attack methods against (Ring-)LWE instances. Remember that when dealing with LWE, we have a dimension n for the samples, a modulus q and an error distribution, say Gaussian with standard deviation σ. We shall see more precisely in the next chapter what values these parameters take in the case of Fan-Vercauteren scheme. Typical values are: few thousands for n, some hundreds of bits for q and σ gets as tiny as 8, despite the hardness reduction that requires $\sigma > 2\sqrt{n}$.

So such parameter set is quite remarkable, q is huge and σ is tiny. Based on this analysis and on the fact that, at the beginning of this thesis, no attacks make any particular assumptions on the parameter values, we started to investigate what could be done in this case and what performance an ad-hoc attack could achieve. This work was mostly done in the first year of my thesis and was accepted to IndoCrypt 2017 as *A Note of Ring-LWE security in the case of Fully Homomorphic Encryption*. Some of the material below is extracted from that work.

3.2.1 Outline

Among the available attacks, some are dedicated to the case where the secret is small, for example BKW [AFFP14] or embedding [BG14] variants. We started by following the path of lattice attacks using decoding approach. This choice stems from the fact that with Fan-Vercauteren application, the public key is one (Ring-)LWE sample and may be attacked offline. Combinatorial attacks would require a huge amount of samples so they seemed less practical. As a running thread for this whole thesis, we shall use Fan-Vercauteren scheme to illustrate our attack.

For the public key to the lattice

From (Ring-)LWE samples it is possible to create a straightforward lattice. We show here how this is done and stress on the main difference between LWE and Ring-LWE cases and why it matters.

In the LWE case, when we have access to m samples of $\mathbb{Z}_q^n \times \mathbb{Z}_q$:

$$\begin{aligned}(\mathbf{a_1}, b_1) &= (\mathbf{a_1}, \langle \mathbf{a_1}, \mathbf{s} \rangle + e_1 \mod q)\\ (\mathbf{a_2}, b_2) &= (\mathbf{a_2}, \langle \mathbf{a_2}, \mathbf{s} \rangle + e_2 \mod q)\\ &\vdots\\ (\mathbf{a_m}, b_m) &= (\mathbf{a_m}, \langle \mathbf{a_m}, \mathbf{s} \rangle + e_m \mod q)\end{aligned}$$

We introduce the matrix $\mathbf{A}$ whose rows are the $\mathbf{a_i}$'s and denote the vectors $\mathbf{b} = (b_1, b_2, \ldots, b_m)$ and $\mathbf{e} = (e_1, e_2, \ldots, e_m)$. The equations above are summed up by the following:

$$(\mathbf{A}, \mathbf{b}) = (\mathbf{A}, \mathbf{As} + \mathbf{e} \bmod q)$$

The second term of the pair $\mathbf{b}$ is, in the q-ary lattice $\mathcal{L}(\mathbf{A})$, the point $\mathbf{As} \mod q$ to which some noise $\mathbf{e}$ has been added.

In the Ring-LWE case, the situation is slightly different. A matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ can be derived from a single sample. With $\mathbf{a} = (a_1, a_2, \cdots, a_n) \leftarrow R_q$ and $\mathbf{s} \leftarrow R_2$, we have the identity $\mathbf{a} \cdot \mathbf{s} = \mathbf{As}$. For easy exposition, if we set n to be a power of two and $R = \mathbb{Z}[X]/(X^n + 1)$, the matrix A is skew-circulant:

$$\mathbf{A} = \begin{pmatrix} a_1 & a_2 & a_3 & \cdots & a_n \\ -a_n & a_1 & a_2 & \cdots & a_{n-1} \\ -a_{n-1} & -a_n & a_1 & \cdots & a_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_2 & -a_3 & -a_4 & \cdots & a_1 \end{pmatrix}$$

The matrix $\mathbf{A}$ differs for other cases of n and R, but the result is the same: with only one Ring-LWE sample, we can mount a lattice attack to attempt to recover $\mathbf{s}$.

Embedding and rescaling

Now that we have a lattice basis $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{Z}_q^n$, we want to find $\mathbf{s}$ and/or $\mathbf{e}$ both in $\mathbb{Z}_q^n$. The attack from Bai and Galbraith goes as follows.

Where is the lattice problem? First, write $\mathbf{A}' = (\mathbf{A}|\mathbf{I_n}) \in \mathbb{Z}_q^{n\times 2n}$. We have the following equality

$$\mathbf{b} = \mathbf{A}' \begin{pmatrix} \mathbf{s} \\ \mathbf{e} \end{pmatrix} \mod q$$

where $\begin{pmatrix} \mathbf{s} \\ \mathbf{e} \end{pmatrix}$ is a short vector, with respect to q, that we want to reveal.

Given $\mathbf{A}$ and $\mathbf{b}$, we can find a trivial solution $\mathbf{w} = \begin{pmatrix} \mathbf{0} \\ \mathbf{b} \end{pmatrix}$ that gives $\mathbf{A}'\mathbf{w} = \mathbf{b}$. However $\mathbf{w}$ is not small, and clearly not the solution we expect. So now that we have a particular solution, we want to find a solution $\mathbf{v_0}$ to the homogeneous problem, i.e. $\mathbf{A}'\mathbf{v_0} = \mathbf{0} \mod q$. For such a vector $\mathbf{v_0}$, we have

$$\begin{aligned} \mathbf{A}'(\mathbf{w} - \mathbf{v_0}) \mod q &= \mathbf{A}'\mathbf{w} - \mathbf{A}'\mathbf{v_0} \mod q \\ &= \mathbf{b} - \mathbf{0} \mod q \\ &= \mathbf{b} \end{aligned}$$

If in addition $\mathbf{v_0}$ is such that $\mathbf{w} - \mathbf{v_0}$ is small, it is most likely that $\mathbf{w} - \mathbf{v_0} = \begin{pmatrix} \mathbf{s} \\ \mathbf{e} \end{pmatrix}$. Consequently, we consider the lattice $\mathcal{L}'$ defined by $\mathcal{L}' = \{\mathbf{v} \in \mathbb{Z}^{2n} : \mathbf{A}'\mathbf{v} = 0 \mod q\}$ and try to find a vector $\mathbf{v_0}$ close to $\mathbf{w}$ in it.

We recognise a CVP instance: $\mathcal{L}'$ is the lattice and $\mathbf{w}$ is the target. We now need a basis for $\mathcal{L}'$ and then we can follow with classical CVP algorithms. Besides, if we know how $\mathbf{s}$ and $\mathbf{e}$ have been sampled, we can derive a bound on $\|\mathbf{w} - \mathbf{v_0}\|$ and this becomes a BDD instance.

Embedding. Luckily, it is fairly straightforward to explicit a basis for $\mathcal{L}'$, by embedding:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I_n} & -\mathbf{A} \\ \mathbf{0} & q\mathbf{I_n} \end{pmatrix} \in \mathbb{Z}^{2n\times 2n}$$

It can be verified that the rows of this matrix are linearly independent and each of them satisfies the definition of $\mathcal{L}'$, so $\mathbf{B}$ is a basis of $\mathcal{L}'$.

This embedding differs slightly from what Bai and Galbraith presented. They instead introduced the matrix

$$\mathbf{M} = \begin{pmatrix} \mathbf{I_n} & -\mathbf{A} \\ & q\mathbf{I_{2n}} \end{pmatrix} \in \mathbb{Z}^{3n\times 2n}$$

and compute its Hermite Normal Form to end up with a full rank matrix generating the lattice. Our technique avoids this computation and yields a basis for $\mathcal{L}'$ more efficiently. We can see that with linear operations on rows their $\mathbf{M}$ can be transformed into our $\mathbf{B}$ as follows:

$$\mathbf{M} = \begin{pmatrix} \mathbf{I_n} & -\mathbf{A} \\ q\mathbf{I_n} & \mathbf{0} \\ \mathbf{0} & q\mathbf{I_n} \end{pmatrix} \to \begin{pmatrix} \mathbf{I_n} & -\mathbf{A} \\ \mathbf{0} & q\mathbf{A} \\ \mathbf{0} & q\mathbf{I_n} \end{pmatrix} \to \begin{pmatrix} \mathbf{I_n} & -\mathbf{A} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & q\mathbf{I_n} \end{pmatrix}$$

Removing the n middle rows gives our matrix. As the name suggests, the Hermite *Normal* Form is useful for theoretical discussions and analyses but an attacker would definitely not go this way. This is why we changed the embedding step.

Rescaling. So far we did not take any advantage of the parameters values, yet in real applications secrets and errors are sampled very differently. Usually $\mathbf{s}$ will be ternary whereas $\mathbf{e}$ is sampled from a Gaussian for standard deviation σ, which in theory should not be small. Consequently the distance from $\mathbf{w}$ to $\mathcal{L}'$ is not balanced along each dimension because $\mathbf{w} - \mathbf{v_0} = \begin{pmatrix} \mathbf{s} \\ \mathbf{e} \end{pmatrix}$.

The idea is to take advantage of that to *inflate* the lattice. If we consider instead a basis

$$\mathbf{B} = \begin{pmatrix} \sigma \mathbf{I_n} & -\mathbf{A} \\ \mathbf{0} & q\mathbf{I_n} \end{pmatrix} \in \mathbb{Z}^{2n \times 2n}$$

the difference between $\mathbf{w}$ and the closest vector in this lattice becomes $\begin{pmatrix} \sigma \mathbf{s} \\ \mathbf{e} \end{pmatrix}$ which is more balanced. It is useful to do this, because now the lattice volume is bigger so lattice reduction algorithms will give better results. It slightly increases the distance between the lattice and the target but it is worth it.

BDD algorithms

Once the embedding and rescaling is done, the next step is to find the closest point $\mathbf{v_0}$ to $\mathbf{w}$ in $\mathcal{L}'$. The available BDD algorithms need a well-reduced basis to work efficiently. This is not the case of our basis $\mathbf{B}$. In the next sections we present our experiments and contributions to the field and see what strategy an attacker would follow.

3.2.2 Expanding on reduction

We already mentioned in the previous chapter that lattice reduction draws many contributions both on theoretical questions and implementation aspects. As we see in this work, the lattice reduction performance and complexity greatly define the security level of a choice of parameters. The previous embedding and rescaling are trivial and costless operations, however breaking BDD is a much more costly task. Since a better reduction allows for a faster enumeration, an attacker would try to balance the time spent in both steps. Hence it is paramount to understand lattice reduction behaviour to provide correct security guarantees.

Ad-hoc algorithms

Based on experimental studies [GN08b, MW16], it seems a natural choice to pick the BKZ algorithm. It achieves quite good reduction, with possible trade-offs on complexity and quality (via the blocksize parameter β). But these experiments are based on general random lattices. The lattice at hand here is very particular: integer, upper triangular, blockwise upper triangular, with scaled identity on the diagonal. The attacker may look for specialised lattice reduction algorithm.

To the best of our knowledge, only one specialised algorithm, a variant of LLL by Gama et al. [GHGN06] may be useful in our case. It is a study about NTRU attack and the lattices

that the authors aim at reducing are *symplectic*. A lattice with basis $\mathbf{B}$ is symplectic if and only if

$$\mathbf{B}^T J_{2n} \mathbf{B} = J_{2n}, \quad \text{where } J_{2n} = \begin{pmatrix} \mathbf{0} & \mathbf{I_n} \\ -\mathbf{I_n} & \mathbf{0} \end{pmatrix} \in \mathbb{Z}^{2n \times 2n}$$

The bases of our case are symplectic, as can be verified. So this variant of LLL would be interesting to try. The authors claim it brings a speed-up factor of nearly 10 when compared to reference implementations of regular LLL. However the code has not been released, so we could not use it in our tests, nor confirm its performance.

This is one example of specialisation that could bring unexpected computational advantage to an attacker. Despite our efforts we were not able yet to come up with another reduction method for this specific case of embedding. The size of the lattice being doubled by the embedding, the reduction time is drastically increased. Consequently working on the original lattice $\mathbf{A}$ before embedding might give a fantastic gain compared to what we present below. We leave this topic for further research.

Tweaking LLL

After a few toy examples we realized that most of the time was spent in the lattice reduction stage. So we tried different settings for BKZ and also for LLL on cases where $n \leq 100$. It turned out that LLL reduction was good enough for our instances, and even a *weak* LLL reduction.

Concerning BKZ, we could confirm [CN11], an intermediate blocksize $\beta = 20$ leads to moderate running times. Whereas smaller values like 5 or greater like 40 lead to higher running times.

Concerning LLL, we recall the size reduction and Lovász conditions:

$$\forall i < j, \quad |\langle \mathbf{b_j}, \mathbf{b_i^\star} \rangle| \leq \eta \cdot \|\mathbf{b_i^\star}\|^2 \iff \mu_{i,j} \leq \eta$$
$$\forall i, \quad \delta \|\mathbf{b_i^\star}\|^2 \leq \|\mathbf{b_{i+1}^\star}\|^2 + \frac{(\mathbf{b_{i+1}} | \mathbf{b_i^\star})^2}{\|\mathbf{b_i}^\star\|^2}$$

We always have $1/4 \leq \delta \leq 1$ and $1/2 \leq \eta \leq \sqrt{\delta}$ and the default value is $(\delta, \eta) = (0.99, 0.51)$.

We first tried to decrease δ to loosen the Lovász condition. However with an LLL-(0.75, 0.51), we see speed-ups only for $n \geq 100$ and the overall attack fails later due to a lattice basis of not sufficient quality.

So we then worked on η, trying to increase its value while keeping $\delta = 0.99$. We performed experiments with the following η: 0.60, 0.75, 0.80, 0.85, 0.95, 0.98. As we can see in Figure 3.1, loosening the size reduction condition decreases the running time of LLL. We observe a speed-up even in moderate dimensions. This speed-up is greater between $\eta = 0.60$ and $\eta = 0.75$ than between $\eta = 0.75$ and $\eta = 0.95$. In addition, with $\eta = 0.95$ we get a less successful attack, whereas when η equals 0.50 or 0.75 the success rates are similar. We conclude that the best option is around $\eta = 0.60$.

Consequently, in our specialised attack, the best choice for the lattice reduction step is an LLL reduction, with $\delta = 0.99$ and $\eta \in [0.60, 0.75]$.

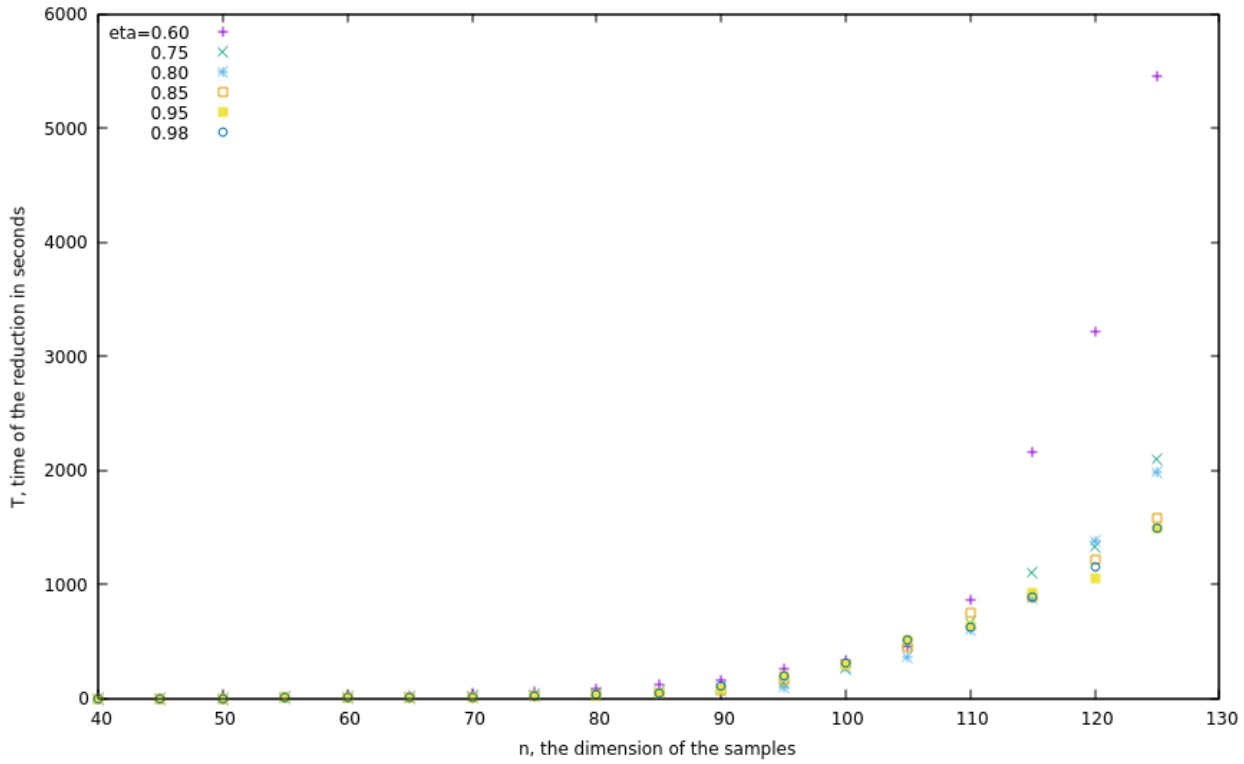


Figure 3.1 – Execution time in seconds in term of n for different η

3.2.3 The actual decoding step

To perform the final search for the closest vector of $\mathbf{v_0}$ in the lattice $\mathcal{L}'$, we have several options. There are sieving algorithms [BTS16, HK17] and enumeration algorithms [FP85, GNR10], with also Gaussian sampling [ADRS15, ADSD15] or Voronoi cell computation [MV10] techniques.

From an attacker's point of view, today the enumeration algorithms are best, especially what is called *pruned enumeration* [LN13]. This method is an adaptation to BDD of the extreme pruning technique introduced by Gama et al. [GNR10]. Conceptually, enumeration is about trying many linear combinations of basis vectors to find vectors that are small (in the case of SVP) or close to the target (in the cases of CVP or BDD). If we do not put any constraints, enumeration generates *all* the lattice vectors. This is a sure way to find the one we look for, but unimaginably costly. So in practice, we want to enforce some condition to keep the *enumeration tree* size under control.

The most elementary way to perform an enumeration comes from Babai [Bab86] *Nearest Plane* algorithm. Assume we have our reduced basis $\mathbf{B} = (\mathbf{b_1}, \cdots, \mathbf{b_n})$ and we want to approximate $\mathbf{w} \notin \mathcal{L}'$. We can express exactly $\mathbf{w}$ with a *real* linear combination of the Gram-Schmidt vectors:

$$\mathbf{w} = \sum_{i=1}^{n} t_i \mathbf{b_i^\star}, \text{ where } \forall i, t_i \in \mathbb{R}.$$

So if we iteratively round each t_i, and update the target to account for the rounding at every dimension, we can get a vector $\mathbf{v} \in \mathcal{L}'$ rather close to $\mathbf{w}$ in polynomial time. More precisely, $\mathbf{v}$ is the unique vector such that $\mathbf{w} - \mathbf{v} \in \mathcal{P}_{1/2}(\mathbf{B}^\star)$. However $\mathbf{v}$ is not always the closest vector to $\mathbf{w}$, so Lindner and Peikert [LP11] generalised this algorithm into the *Nearest Planes* algorithm. In this version, instead of keeping only the rounded coefficients, a bounded exhaustive search is performed to keep more than one candidate at each dimension.

The next level of generalisation is what we call pruned enumeration. This time, at each depth in the tree (i.e. when considering each component of $\mathbf{w}$), we want to choose a candidate component for v_i the i-th coordinate of the solution $\mathbf{v}$ we search. So we look at the quantity $\langle \mathbf{w} - \mathbf{v}, \mathbf{b_i^\star} \rangle / \|\mathbf{b_i}^\star\|$, and consider all values for v_i such that this quantity remains below some bound R_i. We call the tuple $(R_1, \cdots, R_n)$ the *pruning function*. This is the only setting for the last attack step.

In our case, we know precisely the expected distance between $\mathbf{w}$ and the lattice point $\mathbf{v_0}$, $\mathbf{w} - \mathbf{v_0} = \binom{\sigma \mathbf{s}}{\mathbf{e}}$, so we can set the bound R_n to the expected value of its norm. For a Gaussian error distribution we have with high probability $||\mathbf{e}||^2 \leq n \times (3\sigma^2)$, so $R_n^2 = n\sigma^2 + n(3\sigma)^2 = 10n\sigma^2$. It remains to explore how to set the other R_i's. Gama et al. [GNR10] study different remarkable functions: linear, step bounded or piecewise linear. Their results is that *extreme pruning* is the most efficient regime. Extreme pruning is about having very low bounds R_i so that the enumeration tree is very small and quickly scanned. In this case however, the probability of success for finding the correct solution is very low, so we need to shuffle the basis and start again lattice reduction and enumeration multiple times.

Despite this state-of-the-art, we noticed with our instances that the correct solution was almost always the *Babai solution*, that is the first candidate to be enumerated. We provide explanation for this fact below.

3.2.4 Launch the attack, for real

In order to practically evaluate the performance of the ad-hoc attack described above, we implemented it and ran many experiments against real instances, leading to successful breaks.

Early experiments

Our first rounds of experimentation were done using a custom implementation in C++. We made use of the following libraries:

- Lepoint's implementation [Lep14] of Fan-Vercauteren. At that time (2015), SEAL [LCP17] had not yet shifted to FV, but today we would rather use SEAL instead.
- `fplll` [dt17] for lattice reduction algorithms,
- NTL [Sho] for additional lattice operations.

We implemented the embedding/rescaling, the pruned enumeration algorithm from Liu-Nguyen following their pseudo-code [LN13] and created the glue between the different libraries to lead the attack from beginning to end.

Lepoint's implementation. In order to compare FV and YASHE [LN14], Lepoint implemented the code needed to use FV scheme and perform homomorphic operations. We use his constructor method with light modification. It allows us to create a public key `pk` $= ([-(\mathbf{a}\cdot\mathbf{s}+\mathbf{e})]_q, \mathbf{a})$, given n, σ and q. The attack then works with an `FVKey` object as input.

We run these experiments against FV public keys of fairly wide ranges of parameters: $30 \leq n \leq 360$ and $10 \leq q \leq 2^{128}$ keeping $\sigma = 2\sqrt{n}$ (i.e. best choice for FHE, within the LWE hardness bounds). For lattice reduction, we kept $\eta \in \{0.51, 0.61, 0.71\}$.

With n up to 250 we see in Figure 3.2 that with $\eta = 0.71$ the attack takes fairly less time than 0.51, roughly 5 times less and still finishes successfully. This motivated us to keep only the version $\eta = 0.71$ and continue to higher dimensions.

In the end we were able to successfully break an FV key with $n = 320$ and $\log q = 68$ in little less than 29 hours, see Figure 3.3. It seems in reach to beat the previous record from Laine and Lauter [LL15]. They were able to recover a key in dimension 350 in 3.5 days, with a less generic attack with $\log q = 52$ and $\sigma = 8/\sqrt{2\pi}$.

Here we say that we were successful in our attack, meaning we successfully recovered the private key from the public key with the settings we discussed above: LLL reduction with $\eta = 0.71$, $\delta = 0.99$ and Babai enumeration, i.e. the attack is only made of polynomial time components. If it would scale, it would have been an issue for lattice-based homomorphic encryption schemes. However, beyond the successful instances we also had failures, for smaller q, and/or bigger n. Therefore we made thorough analyses of success/failure cases, to push this polynomial time attack as far as possible.

We distinguish several situations:

- *Wrong* when the attack goes as planned and output a vector, but not the correct secret,

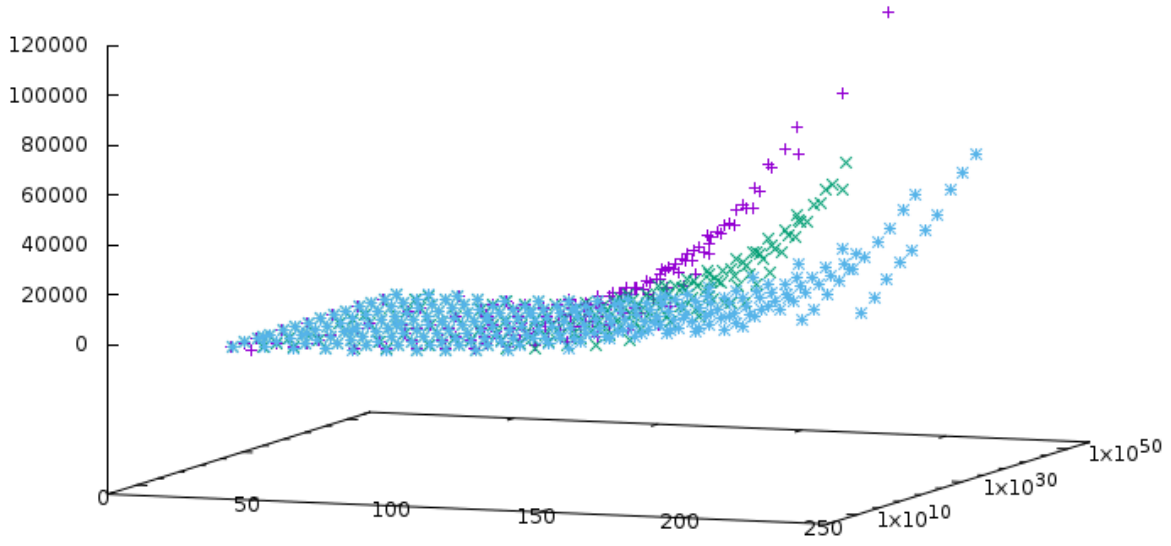


Figure 3.2 – Execution time in seconds in terms of n and q for different η

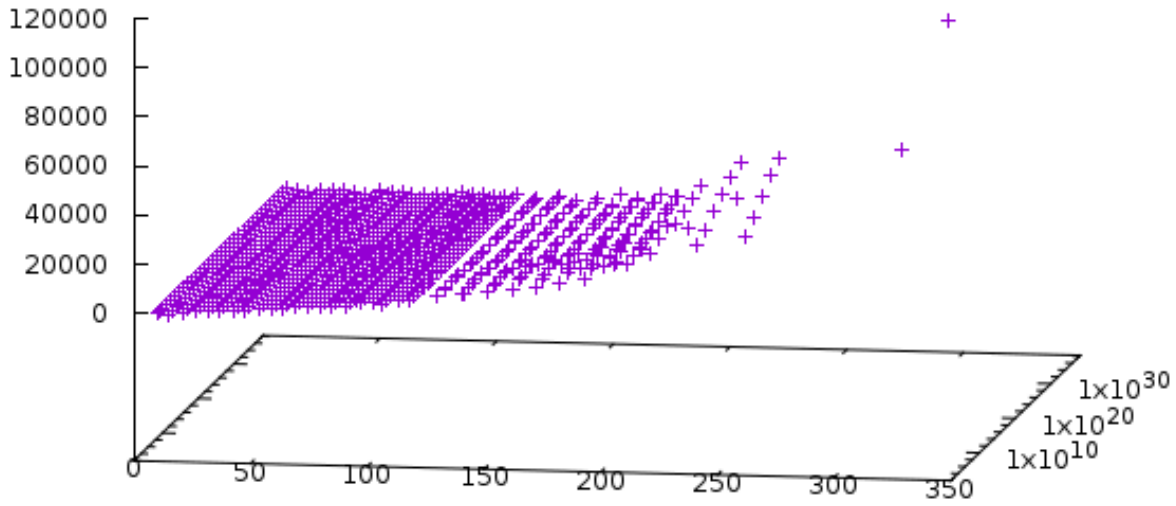


Figure 3.3 – Execution time in seconds in terms of n and q for different $\eta = 0.71$

- *Failure* when Babai enumeration does not succeed in finding a candidate of expected norm.

Looking at lattice reduction indicators, as shown in Figure 3.4, was unfortunately not enough to divide Success from Failure cases.

Decoding from `fplll`

Later, during our study of Success and Failure cases, we dived into `fplll` source code and found that, in addition to the reductions and SVP routines, there was also a CVP routine, flagged experimental. We defer to the next section the details about the work on `fplll` that followed. Soon enough we were able to replace our own implementation of pruned enumeration with a respected library alternative.

So, we launched again the attack on the same instances as before, for the sake of comparison. It turned out that the results were almost the same, in terms of Failure or Success. The only improvement with `fplll` version was on small dimension and huge modulus cases of Failure, which all became Success, probably thanks to a more clever floating-point precision management in `fplll`.

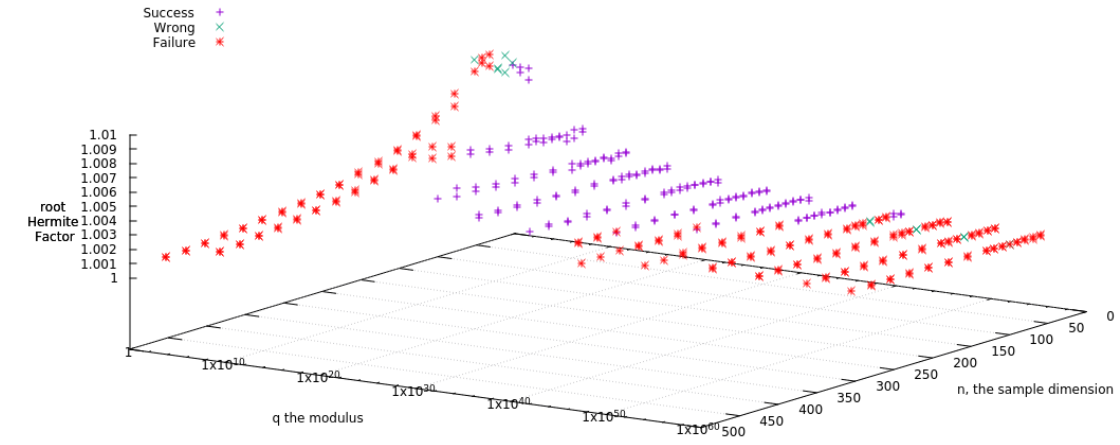
Looking at Babai's Nearest Plane algorithm provides an explanation of the ceiling our attack faces. No successful break happens above $n = 340$. Indeed, we said that Babai algorithm recover the unique vector $\mathbf{v}$ such that $\mathbf{w} - \mathbf{v} \in \mathcal{P}_{1/2}(\mathbf{B}^\star)$, so our attack is successful if and only if $\begin{pmatrix}\mathbf{s}\\ \mathbf{e}\end{pmatrix} \in \mathcal{P}_{1/2}(\mathbf{B}^\star)$. In other words, with a fixed lattice reduction, the $\mathbf{b}_\mathbf{i}^\star$ are also fixed and only an error $\mathbf{e}$ that validates $|\langle \mathbf{e}, \mathbf{b}_\mathbf{i}^\star \rangle| < \|\mathbf{b}_\mathbf{i}^\star\|^2/2$ can be handled successfully. In the LWE case, it is possible to derive a success probability for the Nearest Plane algorithm, depending on the standard deviation of the error and the norm of the Gram-Schmidt vectors [LP11]:

$$P = \prod_{i=1}^{n} \operatorname{erf}\left(\frac{\|\mathbf{b}_\mathbf{i}^\star\|\sqrt{\pi}}{2\sigma}\right)$$

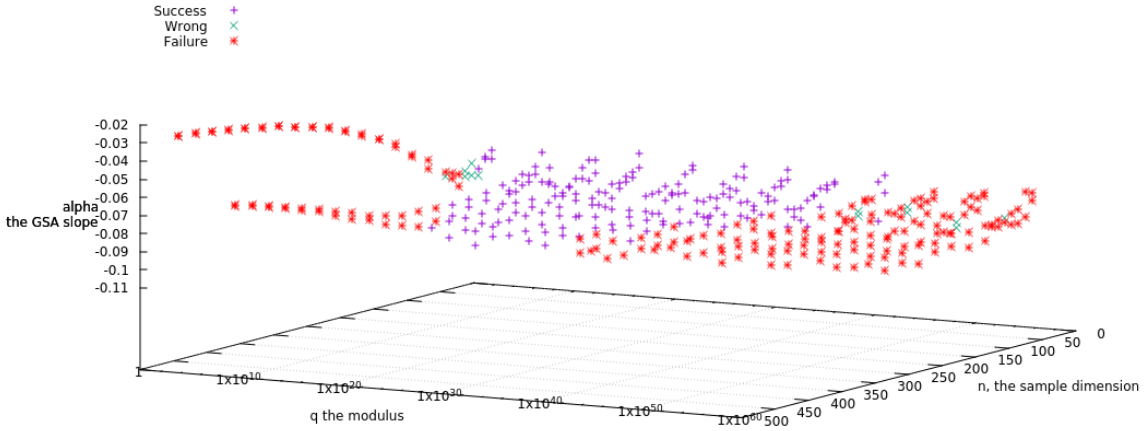
where erf is the Gauss error function. We can exhibit this graphically. In Figure 3.5 we plot the Gram-Schmidt norms of the bases vectors, for reduced lattices of increasing sizes. We also plot the expected error sizes for these dimensions. We can see that, with an increasing dimension, the norm of the smallest Gram-Schmidt vector get smaller while the error norm gets bigger. So obviously, for a fixed reduction, there is a maximal dimension beyond which Babai algorithm cannot be successful.

Easier from Sage

Much later, with the knowledge that this polynomial time attack was limited, we reconsidered the lattice reduction step, in order to push further the successful breaks. Our motivation stems from the fact that third generation homomorphic encryption schemes work with much small dimensions than the second generation (one thousand instead of many thousands). For the new round of experiments, we moved to SageMath environment. It gives access to the efficient `fplll` core routines, thanks to the Python wrapper `fpylll` [Alba] and allows for



(a) root Hermite Factor after reduction



(b) GS slope after reduction

Figure 3.4 – Success, Wrong and Failure cases

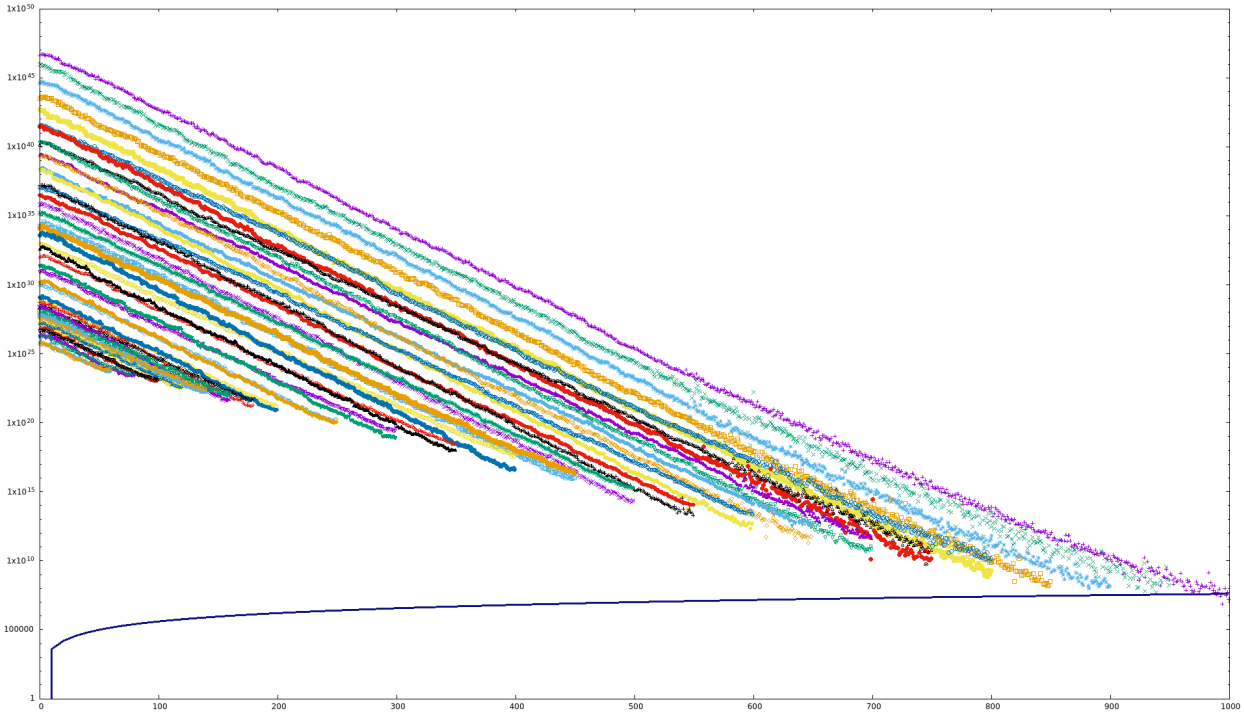


Figure 3.5 – Gram-Schmidt norms and error norms

Sample dimension	100	125	150	175
Lattice dimension	200	250	300	350
Algorithm	LLL-(0.71, 0.99)	LLL-(0.71, 0.99)	LLL-(0.61, 0.99)	BKZ-20
Time (in sec)	78	482	1,705	10,023

Table 3.1 – Successful breaks of SHIELD with diminished dimension

scripting/developing in Python. This is a very comfortable setup to experiment on lattice reduction. Sage also includes Albrecht's `lwe-generator` [Albc].

Our ultimate goal was to come close to the parameters of SHIELD [KGV16]. For 80 bits of security they give: $n = 1024$, $\log q = 31$ and $\sigma = 10$. In the end we could not go very far. We report in Table 3.1.

Conclusion

In this chapter we presented our work on the construction of a dedicated attack to the LWE instances of homomorphic encryption. This has been published at IndoCrypt 2017 [BF17]. At the time of this study, there were no similar work and people were expecting our outcomes. Despite its minor extent, our work has set a first stone to special-purpose attacks against homomorphic encrytion. The state-of-the-art today is the work from Albrecht [Alb17]. We think we can expect many other specific works like ours and we need them to gain confidence

in homomorphic encryption.

This first work on attack made us dive into the state-of-the-art on cryptanalysis of lattice based cryptography. Among others we discovered lattice reduction challenges which are addressed in the next chapter.

Throughout this cryptanalytic work we grasped the wide range of theoretical and practical results. The spread of outcomes is such that one can easily get lost. This contribution has really been a hands-on work and it served as a good start to address our next topics with more confidence.

Chapter 4

Contributions to lattice reduction

4.1 Review of reduction algorithms

A central family of algorithms in our discussions is that of lattice reduction algorithms. In order to manipulate a lattice, we usually handle a family of vectors that spans the lattice, the basis. Therefore the legitimate question is whether there are more efficient representations (i.e. *better* bases) than others, and to what criteria. Then the challenge will be to somehow *improve* the basis.

As shown in Figure 4.1, the second basis seems much better than the first. The experimental intuition we have is that the vector lengths are more balanced and the vectors are closer to orthogonal from one another. While this is trivial to grasp and achieve in dimension 2, it is far from being the case in higher dimensions.

The operation of taking a basis from a lattice and making changes to it in order to improve its quality is called lattice reduction. We present here the different algorithms from

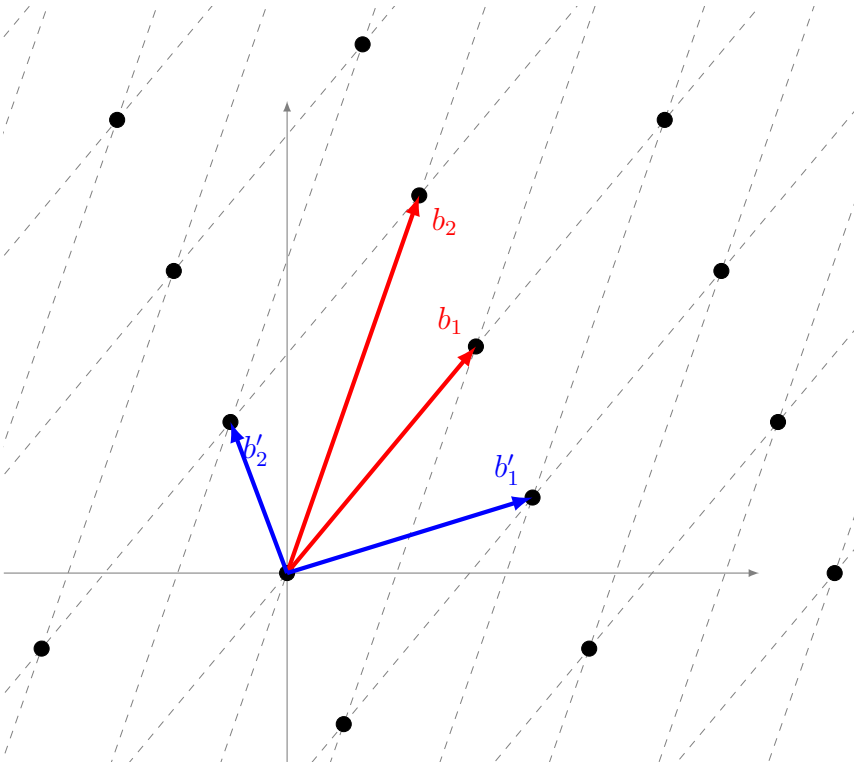


Figure 4.1 – Reducing the bad (red) basis (b_1, b_2) to a better (blue) basis (b'_1, b'_2)

the literature and the performances. Many of the criteria involve the Gram-Schmidt vectors as defined above.

4.1.1 Size reduction

The most elementary reduction is about balancing the sizes of the vectors. We say a basis $\mathbf{B} = (\mathbf{b_1}, \ldots, \mathbf{b_n})$ is size reduced with parameter $\eta \geq 1/2$ if

$$\forall i < j, \quad |\langle \mathbf{b_j}, \mathbf{b_i^\star} \rangle| \leq \eta \cdot \|\mathbf{b_i^\star}\|^2 \iff \mu_{i,j} \leq \eta$$

The algorithm to achieve size-reduction is very simple and runs in polynomial time. It consists, for each vector, in subtracting the component of the current vector with respect to the previous vectors.

Unless specified, the parameter is set to $\eta = 1/2$ in the literature. Though we can see that the smaller the η, the tighter the inequation and the better the reduction becomes.

4.1.2 Lenstra-Lenstra-Lovász (LLL) reduction

The celebrated work from Lenstra, Lenstra and Lovász [LLL82] presents a better lattice reduction algorithm dubbed LLL. It produces a basis that is size reduced, and validates the so-called *Lovász condition*:

$$\forall i, \quad \delta \|\mathbf{b_i^\star}\|^2 \leq \|\mathbf{b_{i+1}^\star}\|^2 + \frac{\langle \mathbf{b_{i+1}}, \mathbf{b_i^\star} \rangle^2}{\|\mathbf{b_i}^\star\|^2}$$

LLL algorithm also runs in polynomial time in the input basis. Both the dimension and the entry size matter for the complexity and for the implementation. Indeed, implementing a floating-point version of LLL is a real challenge and many efforts have been done in this area.

The parameter δ can be tweaked to achieve different output quality. The default value for LLL implementation is $\delta = 0.99$. We shall always keep $\delta \in]1/4, 1]$ and $\eta \in [1/2, \sqrt{\delta}[$.

4.1.3 Blockwise algorithms

As generalisation of LLL, whose Lovász condition considers only a vector and the next at the same time, a new family of algorithms working on *blocks* was developed.

In BKZ (Blockwise Korkine-Zolotarev) algorithm [SE94], the basis is first LLL-reduced for some η and δ. Then the algorithm works on a block of size β. The block is the space spanned by the projections of $\mathbf{b_j}, \ldots, \mathbf{b_{j+\beta}}$ orthogonally to $(\mathbf{b_1}, \ldots, \mathbf{b_{j-1}})$. In this block, the algorithm searches for a short vector, then inserts it in the local basis and call LLL on the new expended vector family to keep β vectors. Thus a BKZ-reduced basis is LLL-reduced and we have the extra property that:

$$\forall 1 \leq j \leq n, \quad \|\mathbf{b_j^\star}\| = \lambda_1(\mathcal{L}_{[j,k]})$$

where $k = \min(n, j + \beta - 1)$ and $\mathcal{L}_{[j,k]}$ is the lattice spanned by $(\pi_j(\mathbf{b_j}), \ldots, \pi_j(\mathbf{b_k}))$. In the case where $\beta = n$, we speak about Hermite Korkine-Zolotarev (HKZ) reduced basis.

While HKZ reduction is a useful notion in theory, in practice it is never attempted because of the time it would need. BKZ, as described here, is no longer used either. Several heuristics have been introduced to improve its performance and reduce its execution time. All these improvements are dubbed BKZ 2.0 [CN11] and we shall present our further improvements later. BKZ 2.0 is the state-of-the-art in terms of time and quality.

Self-dual BKZ. A new approach has been presented by Micciancio and Walter [MW16]: Self-Dual BKZ (SDBKZ). In BKZ, each tour look one block after the other towards increasing indices and extract a short vector in each block. In addition to these *forward* tours, they perform *backward* tours looking at the dual of each block while going back towards the first vectors. The experimental time and quality of SDBKZ are similar to that of BKZ, yet the theoretical analysis is much easier. It provides a more confident way to extrapolate lattice reduction performance to bigger dimensions, which is paramount to establish security level as we shall see below.

4.1.4 Slide reduction

When proving the output quality of LLL, one recovers inequalities of the same flavour as that of Hermite's inequality which states that $\gamma_n \leq \left(\sqrt{4/3}\right)^{n-1}$. Therefore as Gama and Nguyen put it [GN08a], LLL can be viewed as an algorithmic version of Hermite's inequality. This suggests that given another inequality on γ_n, it could be possible to derive its algorithmic

version. This is exactly what they did with Mordell's inequality, which gives

$$\gamma_n \leq \gamma_k^{\frac{n-1}{k-1}} \quad \text{for } 2 \leq k \leq n$$

The resulting algorithm is called Slide Reduction and is another blockwise generalisation of LLL. It also makes a polynomial number of short vector searches and its theoretical analysis is easier than that of BKZ. However the first experimental results did not show attractive speed-ups and it was not used until Micciancio and Walter [MW16] reinstated it as a good reduction algorithm.

4.1.5 Performances and output qualities

Before comparing the algorithm, we shall have some common quality measure(s).

How to measure a basis quality?

Despite the very visual insight we have in dimension 2, as in Figure 4.1, it is not trivial to have such a clear indicator in arbitrarily large dimensions. We recall here the different criteria used in the literature.

Gram-Schmidt norms and the Geometric Serie Assumption. We saw that the Gram-Schmidt (GS) vectors $\mathbf{b_i^\star}$ and coefficients $\mu_{i,j}$ play central roles in lattice reduction algorithms. So it is quite natural to look at these vectors for reduced bases, especially their norms $\|\mathbf{b_i^\star}\|$. In a log-plot, as in Figure 3.5, we see that the decreasing curves can be approximated by a line. This is referred to as the *Geometric Series Assumption (GSA)* [Sch03]

$$\forall i \in \{1, \ldots, n\}, \quad \|\mathbf{b_i^\star}\| = \alpha^{i-1}\|\mathbf{b_1}\|, \text{ for some } \alpha \in [0, 1]$$

We can see throughout a reduction (between BKZ tours for example) that reducing more makes the line closer to the horizontal. α gets closer to 1 and would be 1 in the case of an orthonormal basis.

Root Hermite factor. Without question the most popular indicator is the root Hermite factor, usually noted δ_0 it is defined by

$$\|\mathbf{b_1}\| = \delta_0^n \cdot \mathrm{Vol}(\mathcal{L})^{1/n}$$

It essentially captures how small the first vector of the basis is. However, under the GSA, it conveys information on the whole basis. Indeed if we combine the equations, we get

$$\begin{aligned}\mathrm{Vol}(\mathcal{L}) &= \prod_{i=1}^{n} \|\mathbf{b_i}\| = \prod_{i=1}^{n} \alpha^{i-1} \|\mathbf{b_1}\| \\ &= \prod_{i=1}^{n} \left(\alpha^{i-1} \cdot \delta_0^n \cdot \mathrm{Vol}(\mathcal{L})^{1/n}\right) \\ &= \left(\prod_{i=1}^{n} \alpha^{i-1}\right) \cdot \delta_0^{2n} \mathrm{Vol}(\mathcal{L})\end{aligned}$$

Hence, $\alpha^{n(n-1)/2} = \delta_0^{-2n}$, which yields $\alpha \approx 1/\delta_0^2$. Informally, this means that under the GSA, making the first basis vector shorter yields a better basis. Consequently we usually use this root Hermite factor, quite easy to compute, to measure the (whole) basis quality.

The orthogonality defect. Another measure that takes into account the whole basis is the orthogonality defect. This one is about comparing the volume of the lattice with the product of the basis vectors lengths.

$$od(\mathcal{L}) = \frac{\prod_{i=1}^{n} \|\mathbf{b_i}\|}{\mathrm{Vol}(\mathcal{L})} = \frac{\prod_{i=1}^{n} \|\mathbf{b_i}\|}{\prod_{i=1}^{n} \|\mathbf{b_i^\star}\|}$$

We can see that $od(\mathcal{L}) \geq 1$ with equality when the basis is orthogonal.

Half-volume. This other measure captures the balance between the two halves of the basis. It is defined by

$$V_{1/2}(\mathcal{L}) = \frac{\prod_{i=1}^{\lfloor n/2 \rfloor} \|b_i^\star\|}{\prod_{i=\lfloor n/2 \rfloor + 1}^{n} \|b_i^\star\|}$$

This one is more difficult to compute but can be useful to estimate enumeration times in the reduced basis. Again we would want it to be closest to 1.

Algorithms' complexities and performances

We summarize here the costs of the algorithms presented above, together with the root Hermite factor they achieve. Some results come from proven bounds and theoretical results, other from practical experimental results. When speaking about complexity, we have n the dimension of the lattice (or the basis matrix, we keep the full-rank case) and B is an upper bound of the entries of the matrix, i.e. $\log B$ is the size of the entries.

LLL. The initial LLL proposal has a complexity of $O(n^6 \log B)$ and achieves a theoretical quality of $\delta_0 = (4/3)^{\frac{n-1}{4n}}$ [LLL82]. In other words, it is a polynomial time

algorithm that achieves an exponential approximation factor. After many improvements [NS05, NS09, NV09], we now have more efficient algorithms that have complexity $O(n^{5+\varepsilon}\log B + n^{4+\varepsilon}\log^2 B)$. On the implementation side, there are floating-point versions of LLL that run in time $O(n^3 \log^2 B)$ [CN11]. Also, experimental results showed that LLL can achieve quality as good as $\delta_0 = 1.021$, showing sometimes unexpected performance [NS06, GN08b].

BKZ. To estimate the execution time of BKZ, we need to consider extra parameters: the number of tours N required for it to converge and the time t_β of the local search for short vectors. The overall complexity is then $O(N \times n \times t_\beta)$.

While the theoretical upper bound for N is exponential in n [GN08b], heuristic analysis shows that after a polynomial number of tours, namely $N = (n/\beta)^2 \log n$, the output quality is already very close to what is expected [HPS11].

Concerning the cost of the local searches, many studies must be taken into account to provide a correct estimate. There are several approaches to do this: computing the lattice Voronoi cell [MV10], sieving [BTS16, HK17] or enumerating [FP85, GNR10]. Asymptotically, sieving has the lowest complexity, but as of today enumeration with extreme pruning is the most efficient (to break challenges for example). Sieving and enumeration also differ greatly with respect to memory usage. It is polynomial for enumeration but exponential for sieving. Hence we can use the following for t_β

$$t_\beta = \begin{cases} O(2^{\Theta(\beta)}) & \text{for sieving} \\ O(2^{\Theta(\beta^2)}) & \text{for enumeration} \\ O(\beta^{\Theta(\beta)}) & \text{for enumeration with preprocessing} \end{cases}$$

There are other estimates for BKZ running time [LP11, vdPS13], but quite obsolete now that we have a much better understanding of the algorithm.

The theoretical output quality of BKZ is: $\delta_0 = \left(2\gamma_\beta^{\frac{n-1}{2(\beta-1)}+1.5}\right)^{\frac{1}{n}}$. From the work from Chen [Che13], we estimate that the output quality is $\delta_0 = \left(\frac{\beta}{2\pi e}\cdot(\pi\beta)^{1/\beta}\right)^{1/2(\beta+1)}$.

Slide reduction. The theoretical output quality of Slide Reduction is very similar, yet better, than that of BKZ [GN08a]: $\delta_0 = \left(2\gamma_\beta^{\frac{n-1}{2(\beta-1)}}\right)^{\frac{1}{n}}$. In practice they show similar output quality [MW16].

In addition, the theoretical complexity analysis guarantees only a polynomial number of local searches whereas for BKZ this comes only as a heuristic result. In practice Slide Reduction behaves like BKZ.

4.2 Improving reduction algorithms

We saw in the previous section that lattice reduction performance are central to estimating security of lattice-based schemes. Our cryptanalytic work led us to look under the hood

and bring some improvements to lattice reductions, especially around the `fplll` library. We detail below the work on reductions we did in the course of this thesis.

4.2.1 The `fplll` days

The `fplll` library was originated by Stehlé's work a few years ago and is now a collaborative project on GitHub. It brings together several lattice enthusiasts from the cryptology community and is now available from SageMath, thanks to the side project `fpylll`. `fplll` stands for "floating-point LLL", it provides several lattice reduction algorithms: floating-point LLL versions [NS09] which can be easily used through a wrapper that optimises the floating point settings along the computation, BKZ [SE94] algorithm together with BKZ 2.0 [CN11] heuristics and also slide reduction [GN08a] and self-dual BKZ [MW16]. There are also SVP and CVP enumeration routines.

In the same spirit as the Sage Days, the core team of `fplll` has decided to put in place regular meetings (roughly twice a year) to bring together the maintainers for some days. The objective is to leap forward on evolutions with the relevant people present and to give more visibility to the community. I actively took part in the first two sessions of these `fplll` days (June and December 2016), my contributions range from CVP routine enhancement to work on the new heuristics for BKZ and are detailed in this section.

4.2.2 Proven CVP

CVP enumeration is more involved than SVP

For enumeration, the cases of SVP and CVP are very similar. The objective is to enumerate lattice vectors close to $\mathbf{0}$ or some target vector $\mathbf{t}$ but some subtleties arise when doing floating-point implementations and studying carefully their respective complexities. This issue is discussed in [Kan83, Puj08].

Let us remember that we want to find a lattice vector $\mathbf{v} = \sum_{i=1}^{n} v_i \mathbf{b_i} \in \mathcal{L}(\mathbf{B})$ close to some vector $\mathbf{w} = \sum_{i=1}^{n} t_i \mathbf{b_i^\star}$, where the $\mathbf{b_i^\star}$ are the Gram-Schmidt vectors of $\mathbf{B}$. We note $r_i = \|\mathbf{b_i^\star}\|$. We know, assume, or at least want that

$$\begin{aligned}
\|\mathbf{w}-\mathbf{v}\|^2 &= \|\sum_{i=1}^{n} t_i \mathbf{b_i^\star} - \sum_{i=1}^{n} v_i \mathbf{b_i}\|^2 \\
&= \|\sum_{i=1}^{n} t_i \mathbf{b_i^\star} - \sum_{i=1}^{n} v_i \big(\mathbf{b_i^\star} + \sum_{j=i+1}^{n} \mu_{i,j} \mathbf{b_j^\star}\big)\|^2 \\
&= \sum_{i=1}^{n} \big(t_i - v_i - \sum_{j=i+1}^{n} \mu_{j,i} v_j\big)^2 r_i^2 \\
&= \sum_{i=1}^{n} \big(c_i - v_i\big)^2 r_i^2, \qquad \text{with } c_i = t_i - \sum_{j=i+1}^{n} \mu_{j,i} v_j \\
&\leq R^2 \quad \text{for some radius } R \geq 0
\end{aligned}$$

Now the enumeration algorithm works as follows: start at the bottom of the tree $i = n$ and at each depth try all v_i such that

$$(c_i - v_i)^2 r_i^2 \leq R^2 - \sum_{j=i+1}^{n} (c_j - v_j)^2 r_j^2$$

Since we are going toward decreasing indices, v_i is the only variable quantity in the equation above. Also it is an integer, so there is a bounded number of possibilities. We can show that there is a maximum of $2R/r_i + 1$ possible values.

However, if we want to solve CVP and not BDD, i.e. we do not know R in advance, we need to pick a value for it. In the SVP case, setting $R = \lambda_1(\mathcal{L})$ does the trick, but for CVP we need to go to the top of the tree to conclude, which is not the case for SVP. So it is likely that, even for the correct solution, the bound at depth $1 < i < n$ will discard the current candidate. So we need to have at least $R \geq \sum_{i=1}^{n} r_i^2/4$. This bound is not satisfactory either. Indeed, the Geometric Series Assumption tells us that for a reduced basis, the r_i are decreasing. But there are cases, that can be exhibited, for which the suite is arbitrarily increasing. For such case, the present approach that enumerates $2R/r_i + 1$ solution per level is likely to waste enormous amount of time. So we need to do better.

Fixing `fplll`

The algorithmic solution to this is to somehow *reset* the enumeration when we reach a r_i with maximal value before going deeper in the tree. It improves both the running time and the numerical stability (for floating-point implementations). The `fplll` routine for CVP was not using this trick, that is why it was flagged "experimental". During the first `fplll` days, we implemented it with some help from Marc Stevens.

This idea is to separate the depths into the largest intervals in which the r_i are decreasing. Say it is decreasing from $i = 1$ to $i_0 - 1$ for some $i_0 > 1$ and from i_0 to n. Below i_0, it makes sense to set $R = \sum_{i=i_0}^{n} r_i/4 \leq (n - i_0)r_0/4$ but for depth between 1 and $i_0 - 1$, the number of candidates at each depth is probably going to be huge, so it would be better to set the bound differently. At this point, we have some candidates for the last components of $\mathbf{v} = (\star, \ldots, \star, v_{i_0}, \ldots, v_n)$ and we still want to find the firsts. Remember that at this point, our objective is

$$\|\mathbf{w} - \big(\sum_{i=1}^{i_0-1} v_i \mathbf{b_i} + \sum_{i=i_0}^{n} v_i \mathbf{b_i}\big)\|^2 \leq R^2$$

We can view this as a new CVP in dimension $i_0 - 1$ where the new target is $\mathbf{w} - \sum_{i=i_0}^{n} v_i \mathbf{b_i}$. Moreover, instead of going on with a bound $R'^2 = R^2 - \sum_{i=i_0}^{n} (c_i - v_i)^2 r_i^2$, we can pick $R'^2 = \sum_{i=1}^{i_0-1} (c_i - v_i)^2 r_i^2$. In the possible cases where the r_i have different sizes, this approach also improves numerical stability, because where R' would have been of the order of R, i.e. too big, R' depends only on the "small" r_i and will be more adapted.

The code to add this reset feature was integrated to fplll[1]. It involved patching most of the enumeration routines.

[1] `https://github.com/fplll/fplll/pull/191`

4.2.3 BKZ 3

During the second `fplll` days, we worked together with Léo Ducas and Martin Albrecht on new ways to make BKZ reduction faster. We refer to this as *YoloBKZ* or more pompously BKZ 3. We shall now first review the BKZ algorithm in details, together with BKZ 2.0 heuristics and then introduce our new techniques.

Review of BKZ and BKZ 2.0

We briefly said that BKZ was a blockwise algorithm performing several *tours* on the basis, using a shifting window called a *block*. For each position of this block, the local basis is improved with the addition of a newly found short vector. The output quality of BKZ depends mainly on the block-size β. So does its time complexity, which is also dependent on the number of tours. BKZ, was first introduced by Schnorr and Euchner [SE94] and refined by Chen and Nguyen [CN11] who brought several tweaks to make it faster.

More precisely, BKZ algorithm works as described in Algorithm 4.1. We keep a high level description here to focus on the essential aspects. The interested reader can also have a look at an implementation of BKZ in `fpylll` [2]

Algorithm 4.1 Perform a BKZ-β reduction on the input basis $\mathbf{B}$

function BKZ($\mathbf{B}$: an LLL-reduced basis, β: the blocksize)
  repeat
    Tour($\mathbf{B}, \beta$)
  until No progress is made
  return B
end function
function Tour($\mathbf{B}$: the current basis, β, the blocksize)
  for $1 \leq j \leq n$ **do**
    $k \leftarrow \min(j + \beta, n)$
    $\mathbf{v} \leftarrow$FindSVP($\mathbf{B}, j, k$)
    if $\mathbf{v}$ is useful **then**
      insert $\mathbf{v}$ in $\mathbf{B}$
    end if
  end for
end function

Let us know provide more details on the different steps.

No progress is made. The termination condition of BKZ in its original version says that the algorithm should continue as long as the basis is changed during a tour, i.e. a shorter vector is found in at least one of the blocks. It is proven that BKZ terminates, and observed that the number of tours is exponential (in the lattice dimension and size of its entries) [GN08b]. A heuristic result [HPS11] suggests however that it should be enough to perform

[2] `https://github.com/fplll/fpylll/blob/master/src/fpylll/algorithms/simple_bkz.py`

a polynomial number of tours to get an output quality close to final. This is what we call the *early abort* strategy. BKZ is stopped when no "significant" progress is made on a tour. In practice, the GSA slope is monitored and if the progress per tour is below some threshold for some number of tours in a row, the algorithm is stopped.

FindSVP. This SVP oracle may be implemented using techniques we mentioned above: sieving or enumeration. Recent works on sieving [BTS16, HK17, BDGL16] bring it closer to be best in practice, but for a long time enumeration wan out. Most of the heuristics that shape BKZ 2.0 are about optimising enumeration. In `fplll`, enumeration is used, yet the `fplll` days 4 in December 2017 are about bringing in the latest algorithmic improvements on sieving.

With enumeration there is always the question about the enumeration radius. As we are doing SVP we do not have the subtleties of CVP to take care of as in the previous section. We can set $R = \|\mathbf{b}_\mathbf{j}^\star\|$, at least the enumeration will output this vector. Yet the Gaussian Heuristic also provides some clue on how short we expect the vector to be. So in practice, the enumeration radius is set to what is minimal between $\|\mathbf{b}_\mathbf{j}^\star\|$ and 1.05 times the Gaussian Heuristic prediction. Also, as for any enumeration, the pruning function should be chosen to minimize the overall enumeration, even if it implies to perform many cheap enumerations with low success probability on different bases (obtained by randomisation).

Now that we did our best to have a very efficient enumeration, we shall not forget about the quality of the local basis. By design, the BKZ algorithm is such that any block is LLL-reduced when FindSVP is called. LLL-reduced bases offer only fair context to run enumeration, the situation can be improved if we have BKZ-reduced bases. So this is exactly what is done: in the case of a BKZ-β reduction, it is very appropriate to have the local bases BKZ-β' reduced, for some $\beta' < \beta$. Optimising the choices for β' in terms of β has been addressed by Chen and Nguyen when they introduced BKZ 2.0. We shall come back to this later. We also call this step the *SVP pre-processing* step.

`fplll` is distributed with a default strategy that provides "optimised" choices for pre-processing block-sizes, enumeration radiuses and pruning function, depending on the requested BKZ reduction. This strategy shows good results, yet it is established from average reduction behaviour on generic lattices, so it may be sub-optimal for some particular lattices.

If v is useful. The FindSVP oracle should at least return a vector as good as the first of the local basis. If it is the case and not better, there is obviously no need for insertion. BKZ and BKZ 2.0 both share this behaviour.

Insert v into B. When inserting a new vector, the local basis momentarily gets an extra vector. The solution to remove the linear dependencies and to keep the best vectors is to run LLL. This is usually done on the beginning of the whole basis, between indices 1 and $k + 1$. We refer to this insertion and reduction step as the *SVP post-processing* step.

Introducing new heuristics

We present now some new tricks for faster BKZ. This is mainly experimental and not yet published in final versions. It results from manual experiments and tunings and extensively benefits from the flexibility of `fpylll`.

Auto-adaptive pre-processing. Instead of using pre-computed strategies, it would be best to have an adaptive strategy. The same wish extends to all parameters of lattice reduction, of course. Yet for pre-preprocessing Léo Ducas developed some code which monitors the enumeration behaviour (success probability and number of enumerated nodes) to adapt on-the-fly the pre-processing settings that will be used for the next block. We ran several experiments to try to get a trend on how it behaves. This aim was to derive a pre-computed dynamic strategy or strategies. But as we can see in Figure 4.2, it changes completely from one lattice to the other, alternating phases of small and big block-sizes.

In the end, it makes it faster to reach a BKZ 2.0 quality.

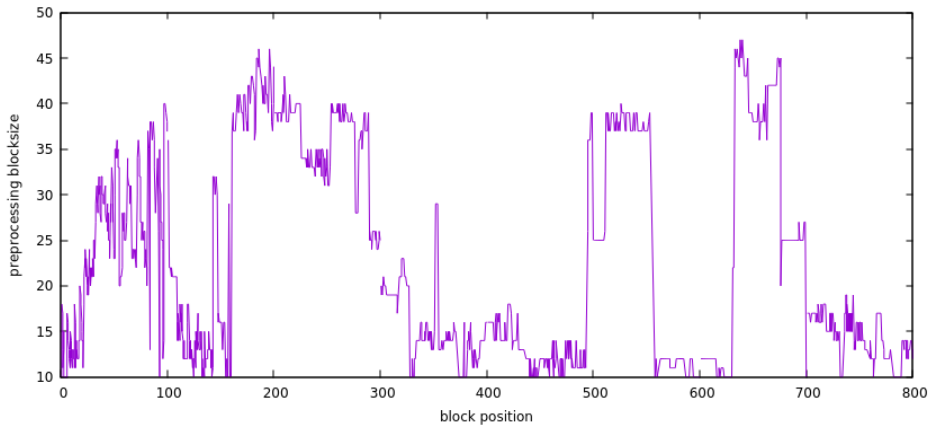
Tentative multiple insertions. The time spent in the FindSVP oracle is a significant part of the total reduction time. Yet each call is worth, at most, one short vector. We investigated the possibility to make use of other candidates, encountered during the enumeration (or sieving). For example, we may wish to insert not only *the* shortest vector found but maybe also the second shortest or more. The post-processing step will automatically handle insertions of many vectors thanks to the LLL behaviour which will keep only β linearly independent vectors. Consequently, we modified the enumeration procedure in `fplll` to enable the return of more than one vector, and tried to do more than one insertion per block.

We explored inserting 2 vectors, 5 vectors, or $\beta/2$ or $\beta/4$. None of these strategies improved the quality or the speed. Dead end.

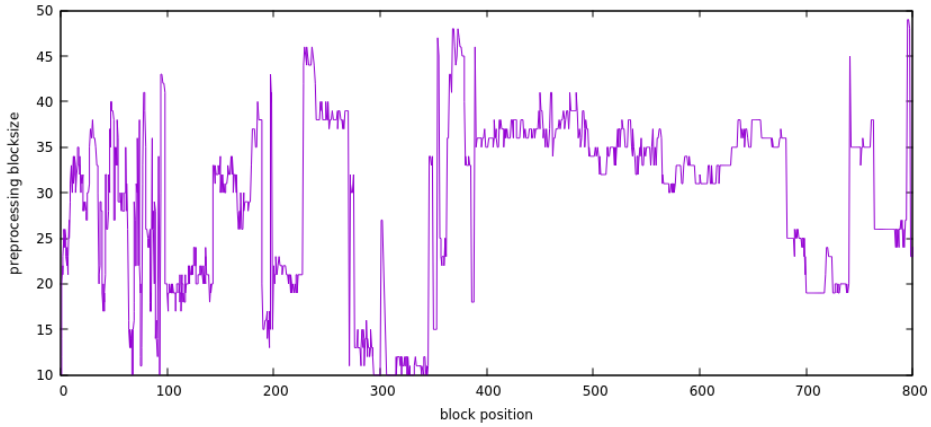
Weaker/less LLL in post-processing. Here we want to decrease the time spent in LLL in the post-processing. First, the vector returned by FindSVP may be easy to insert manually into the basis. So there is some code to handle such cases, like when the vector has only ± 1 coordinates. Next, running a very weak LLL is enough to remove linear dependencies. Though, next time we work in the block, we want it to be reduced enough for the next FindSVP call to work fine. So two options: reducing not to weakly ($\eta = 0.71$ gives good results) or split the work between the current post-processing and the next pre-processing (of this same block, i.e. in the next tour). Second option offers even more possibilities. For example, reducing only parts of the basis before going to the next block, the rest of the work will be done later if needed.

Several leads have been followed independently by Martin, Damien, Léo, Shi, Marc and myself but the techniques are not orthogonal and not necessarily compatible. Below we report some speed-ups between BKZ 2.0 and BKZ including such heuristics. The one we keep from this section is that of using weak LLL with $\eta = 0.71$.

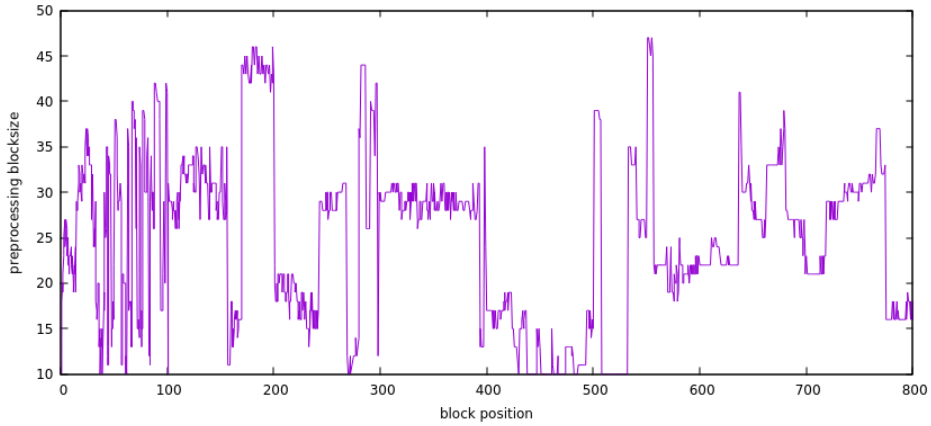
Progressive block-sizes. Similarly as we use locally BKZ with smaller block-size to speed up enumeration in blocks, we may BKZ-reduce the whole basis with a smaller block-size



(a) Random lattice 1



(b) Random lattice 2



(c) Random lattice 3

Figure 4.2 – Optimal choices for pre-processing block-sizes along BKZ-60 tours in dimension 100. From block position 0 to 100 it is the first tour, from 101 to 200 the second and so on.

Blocksize	10	20	30	40	50	55	60
Time (s)	0.198	0.335	0.546	1.24	11.0	24.7	44.9
Cumulative (s)	0.198	0.533	1.08	2.32	13.4	38.1	82.8
$\log \|\|\mathbf{b}_\mathbf{0}^\star\|\|^2$	38.9	38.9	38.7	38.4	37.5	36.7	35.9
GS slope	-0.0851	-0.0830	-0.0791	-0.0725	-0.0643	-0.0575	-0.0523

(a) With ramp-up

Blocksize	60	60	60
Time (s)	54.8	47.5	53.4
Cumulative (s)	54.8	102	156
$\log \|\|\mathbf{b}_\mathbf{0}^\star\|\|^2$	38.1	36.8	36.0
GS slope	-0.0678	-0.0575	-0.0524

(b) Without ramp-up

Table 4.1 – Time and quality improvements of a ramp-up phase

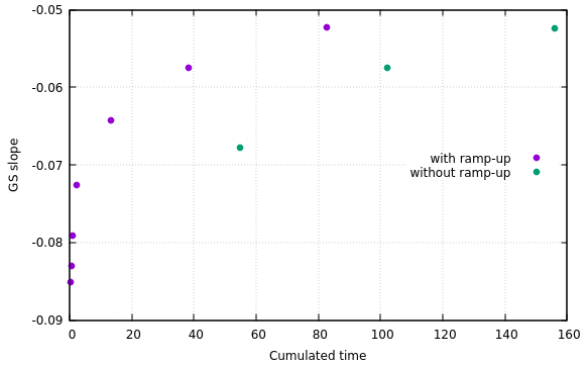
before doing the reduction with the block-size we want. This trend has already appeared in the literature and studied recently [HRP13, AWHT16]. The later work proposes very clever techniques to simulate the best progression toward a target block-size. We adopted a more down-to-earth approach, based on experiments.

Our technique relies solely on a ramp-up phase, prior to the normal BKZ execution. From a practical point of view, on a lattice of dimension 160, BKZ reductions for block-size under 40 are of negligible costs, whereas the closer we get to the target block-size, the less negligible it becomes. So our idea for ramp-up was to jump by a step of 10 until $\beta - 10$ and then $\beta - 5$, doing one BKZ tour for each block-size.

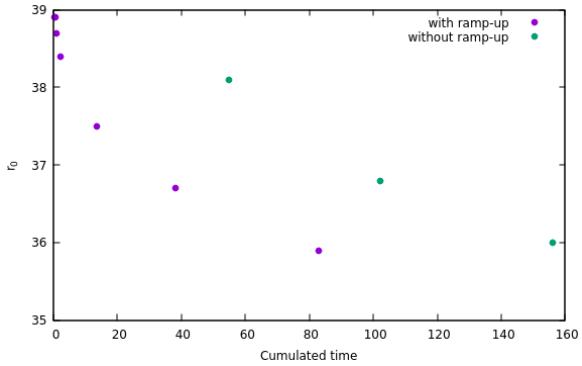
The results are quite impressive (cf. Table 4.1 and Figure 4.3): the ramp-up phase takes about 75 % of one normal tour and the quality after ramp-up and one tour is the same as the quality after doing 3 tours directly. So we have a speed-up from 3 to 1.75 just with this naive technique. Further into the reduction, as the normal tours happen, the benefits of this ramp-up, independently of its nature, fade away. Considering the results, this technique should also be put in place for all local pre-processing BKZ. This would bring speed-ups for the whole BKZ reduction and even in this naive, costless fashion.

4.2.4 2-phases LLL

We made another observation on LLL behaviour during the many experiments we conducted. On average and on our lattices from embeddings, it seems faster to first run a weak LLL reduction, before running the one with our target parameters. We did not investigate a lot, but Figure 4.4 shows that it can make the reduction up to two times faster. Damien Stehlé pointed us that this strategy was already mentioned [Coh96].



(a) GS slope evolution with and without a ramp-up phase



(b) First basis vector norm evolution with and without a ramp-up phase

Figure 4.3 – Basis quality improvement dynamic, with or without ramp-up

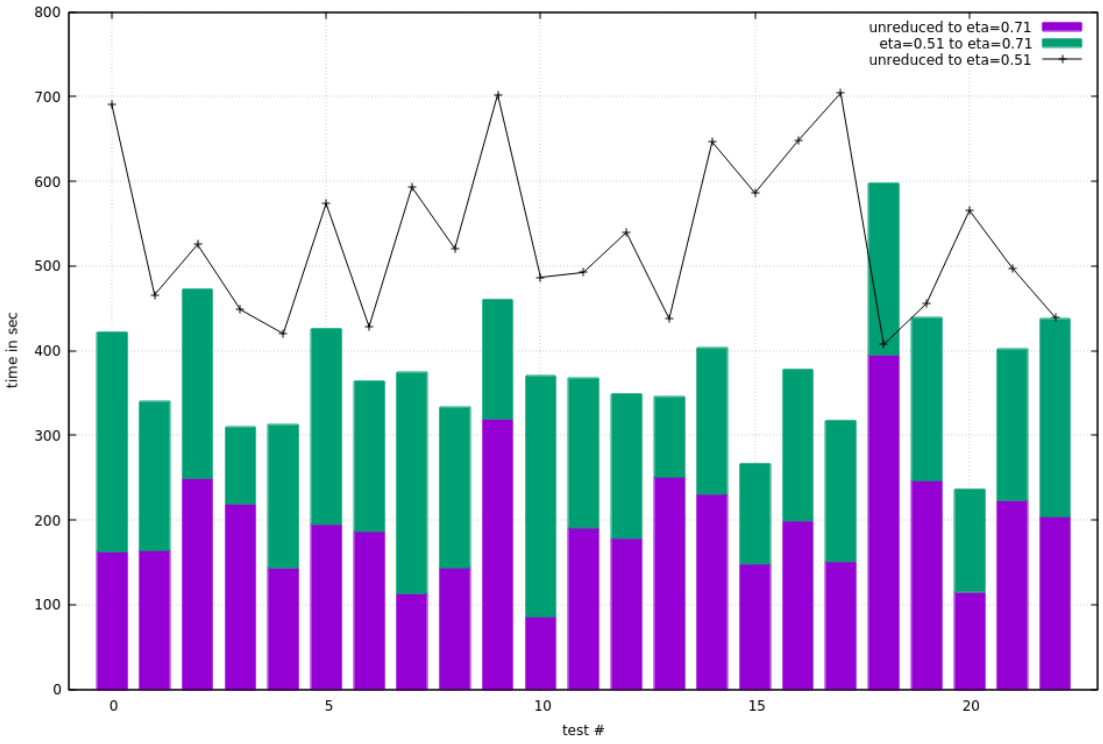


Figure 4.4 – Reduction time, one step versus two steps

Conclusion

We reported here several experimental results on lattice reduction algorithms and our contributions to `fplll`. Interestingly, the ramp-up paradigm seems efficient for both BKZ and LLL algorithm so could be studied more closely. Many heuristics drive the concrete performances of the lattice reduction algorithms, so there are still many hindsights to gain before being able to extrapolate confidently their performances. Also the days of enumeration may soon be over since we see many speedups coming from sieving and its new tricks.

The great discussion within the `fplll` community allowed us to contribute to the long existing field of computational cryptanalysis. Standing between what theorems prove and what computers can do, the landscape of the contribution is of the most exciting. The progresses we helped making bring more confidence to the whole lattice based cryptography community. This is paramount to evolve towards broader acceptation and use of these primitives.

Chapter 5

Comparing and using schemes correctly

Where we demonstrate that the best security is not the highest.

In the beginning of this thesis, we started by working on the cryptanalytic or attack side. Now we can safely head to the task of implementing homomorphic encryption schemes, for real. Considering the state-of-the-art on the existing schemes in 2015, the challenge for us is about choosing parameters for them to ensure both correctness of our computations and enough security. The insights we got from our previous works allow us to confidently address this topic.

In this chapter we shall first review the classical parameter derivation methods. Then, we present a common work with Vincent Migliore whose aim is to provide ready-to-use content for people from outside the cryptography community, e.g. implementers. In the end we also present some results on the influence of the error size over the other parameters.

Most of this work was accepted for a special issue of *IEEE Transactions on computers* on *Cryptographic Engineering in a Post-Quantum World* [**?**].

5.1 Review of the methods

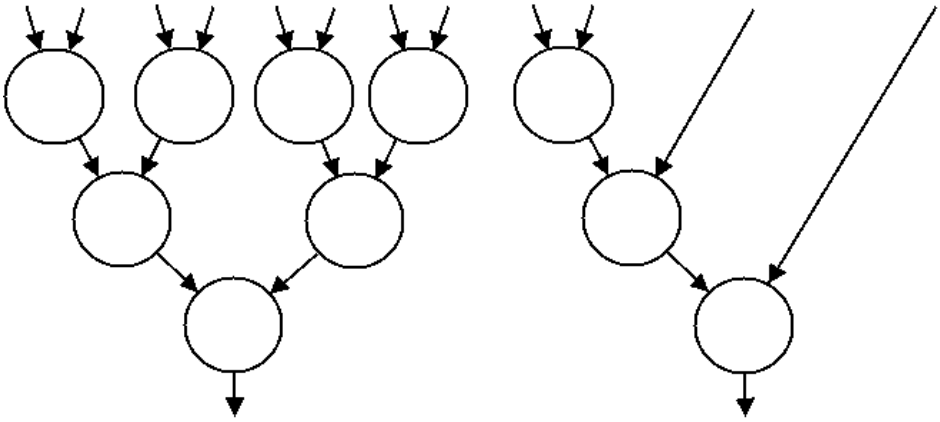
Prior to our study, several works had already be done on scheme analysis and comparison. First, a comparison of FV and YASHE' from Lepoint and Naehrig [LN14]. They derived parameters for both schemes, implemented them in C++ and compared their performances at evaluating a lightweight block cipher SIMON [BTCS+15]. They established that the noise growth in FV is smaller than in YASHE', while the latter turned out to be faster. Later, Costache and Smart [CS16] included BGV. They compared BGV, FV and YASHE' and highlighted the main advantages of each of them, namely BGV is best for large plaintext moduli, and YASHE' wins in other cases.

At the heart of a comparative parameter selection stand two main challenges. The first one is to have a unified description of the schemes, so that when we write equations about the noise growth, i.e. about the correctness of the scheme, we handle comparable data. And secondly, there is a need to draw parameters that achieve the same security level for all schemes.

Addressing the first challenge is *just* a matter of putting the operations into perspective to unify the different actions done to the ciphertexts. Both previous works and ours do that. However when it comes to security, we saw in the previous chapters that huge efforts are being done on the attack front and it can be difficult to confidently draw parameters. As it turns out that YASHE' has been broken, the outcomes of both previous studies should be revised. We remind that [LN14] uses the technique from [vdPS13] and [CS16] uses that of [LP11]. Both techniques attempt to model the power of lattice reduction at solving LWE from only one point of view. However, YASHE' is an efficient variant of YASHE, which was not proven secure, and these models could not account for an attack such as the sublattice attack.

5.2 Comparing FV and SHIELD

Now we present our comparative study of somewhat homomorphic schemes: the schemes included, the use cases. Then we present both FV and SHIELD in our unified presentation and derive their noise growth equations (for scheme correctness) and then we detail the security aspects. The section after documents our findings.



(a) Arbitrary circuit of depth L (b) Optimal circuit of degree L

5.2.1 Scope

Schemes

As we presented earlier in Section 2.4.1, the most practical SHE schemes in 2015 were: BGV [BGV12], FV [FV12], YASHE [BLLN13] (in its optimised version dubbed YASHE'), SHIELD [KGV16] (aka Ring-GSW) and F-NTRU [DS16]. The three first of this list (BGV, FV and YASHE) belong to the second generation of HE schemes while SHIELD and F-NTRU belong to the third.

YASHE and F-NTRU have been damaged by the sublattice attack from 2016 [KF17], so we had originally included them in the scope of our comparative study, but we dropped them later. The attack broke the security assumptions of both YASHE' and F-NTRU. YASHE is still secure but impractical and for F-NTRU the correctness and security constraints can no longer be satisfied.

BGV differs significantly from FV and SHIELD, by its structure involving several moduli. So with hardware considerations in mind, we decided not to include it in our work. Accelerating hardware modular reduction with changing modulus seemed quite a challenge and not promising for performance. Nonetheless, BGV is a good candidate, especially when it comes to *large* plaintext moduli [GHS12b, CSVW16, JA16, CKKS17]. Several works also support the use of binary plaintext. Among other things, it enables more complex operators such as comparison. Hence our choice for FV and SHIELD.

Use cases

Our work aims at comparing FV and SHIELD on different scenarios. The main aspect we consider for comparison is the size of the ciphertexts and key materials, because the costs of the communications and homomorphic operations will then depend directly on them. So we study the parameters in the following cases:

- Evaluation of an arbitrary binary circuit of depth L, i.e. a polynomial for maximum degree 2^L (see Figure 5.1a).

- Evaluation of an optimised binary circuit of degree L (see Figure 5.1b). With SHIELD the noise growth is better when a multiplication occurs with a "fresh" ciphertext (i.e. depth L with one additional multiplication per level). So we highlight the parameters of this optimal case.
- Use of NTT multiplication in the Negative Wrapped Convolution regime. This setting saves a polynomial reduction when doing a multiplication, but implies extra constraints on the dimension n. With high performance objectives it could be interesting to work in this regime, so we study the impact on these extra constraints on the other parameters.
- *Batching* or *Single Instruction on Multiple Data (SIMD)* technique. This trick allows to amortise the computation cost by evaluating the same circuit on several different input data. For this, we also studied the choice of dimensions which yield different numbers of possible slots. This was mostly done by Vincent.

5.2.2 Unified presentation

To begin this comparative work, we present FV and SHIELD (and did also F-NTRU at the beginning) in a somewhat unified formalism.

FV

To detail the brief presentation that we made of FV in a previous chapter, we state in detail the sub-routines of the scheme from Fan-Vercauteren. We note $l_{\omega,q} = \lceil \log_2 q / \log_2 \omega \rceil$. Algorithm 5.1 introduces utility functions, Alg. 5.2 the key generation procedures, Alg. 5.3 the usual encryption and decryption and Alg. 5.4 presents the actual homomorphic elementary operations.

It holds that $\left\langle \text{FV.PowersOf}_{\omega,q}(\mathbf{a}), \text{FV.WordDecomp}_{\omega,q}(\mathbf{b}) \right\rangle = [\mathbf{a} \times \mathbf{b}]_q$.

SHIELD

Now we introduce, under the same formalism, the subroutines of SHIELD. We note $N = 2 \times \lceil \log_2 q \rceil$. For a ring element a we use $a^{(j)}$ to speak about the ring element composed of the j-th bit(s) of a.

Algorithm 5.5 introduces utility functions, Alg. 5.6 the key generation procedures, Alg. 5.7 the usual encryption and decryption and Alg. 5.8 presents the actual homomorphic elementary operations. As the names suggest BitDecompInv is the invert function of BitDecomp. We call them BD and BDI for conciseness.

5.2.3 Noise growth equations

Next, under this unified framework we derive the noise growth equations for both schemes. Indeed, to control correctness one expresses the noise after L multiplications and validates that a decryption of a ciphertext with this amount of noise will yield the expected plaintext. This work has to be done for each scheme.

Algorithm 5.1 FV: Utility functions

function FV.PowersOf$_{\omega,q}(\mathbf{a} \in R_q)$
  $\mathbf{A} = (\mathbf{A_0}, \cdots, \mathbf{A}_{l_{\omega,q}-1}) \in R_q^{l_{\omega,q}}$
  for $i = 0$ to $l_{\omega,q} - 1$ **do**
    $\mathbf{A_i} \leftarrow [\mathbf{a} \cdot \omega^i]_q$
  end for
  return A
end function
function FV.WordDecomp$_{\omega,q}(\mathbf{a} \in \mathcal{R}_q)$
  $\mathbf{A} = (\mathbf{A_0}, \cdots, \mathbf{A}_{l_{\omega,q}-1}) \in R_q^{l_{\omega,q}}$
  for $i = 0$ to $l_{\omega,q} - 1$ **do**
    $l_0 = i \times \log_2 \omega$
    $l_1 = (i+1) \times \log_2 \omega - 1$
    $\mathbf{A_i} \leftarrow \mathbf{a}_{(l_0 \cdots l_1)}$
  end for
  return A
end function

Algorithm 5.2 FV: Key generation functions

function FV.GenKeys$(\sigma_{key}, \sigma_{err})$
  $\mathbf{s} \leftarrow D_{R_q,\sigma_{key}}$
  $\mathbf{a} \leftarrow U_{R_q}$
  $\mathbf{e} \leftarrow D_{R_q,\sigma_{err}}$
  $p_k \leftarrow (-\mathbf{a} \cdot \mathbf{s} + \mathbf{e}, \mathbf{a})$
  $s_k \leftarrow \mathbf{s}$
  return (p_k, s_k)
end function
function FV.GenRelinKeys(p_k, s_k, σ_{err})
  $\mathbf{A} \leftarrow U_{R_q}^{l_{\omega,q}}$
  $\mathbf{E} \leftarrow D_{R_q,\sigma_{err}}^{l_{\omega,q}}$
  $\gamma \leftarrow \left(\left[\text{FV.PowersOf}_{\omega,q}(s_k^2) - (\mathbf{A} \cdot s_k + \mathbf{E}) \right]_q, \mathbf{A} \right)$
  return γ
end function

Algorithm 5.3 FV: Encryption/Decryption

function FV.ENCRYPT($\mathbf{m}, p_k, \sigma_{key}, \sigma_{err}$)
  $\mathbf{u} \leftarrow D_{R_q,\sigma_{key}}$
  $(\mathbf{e_1}, \mathbf{e_2}) \leftarrow D^2_{R_q,\sigma_{err}}$
  $\mathbf{c} \leftarrow \left(\left[q/t \times \mathbf{m} + p_{k,0}\mathbf{u} + \mathbf{e_1} \right]_q, \left[p_{k,1}\mathbf{u} + \mathbf{e_2} \right]_q \right)$
  return c
end function
function FV.DECRYPT($\mathbf{c}, s_k$)
  $\widetilde{\mathbf{m}} \leftarrow \left[\mathbf{c}_0 + \mathbf{c}_1 \cdot s_k \right]_q$
  $\mathbf{m} = \lfloor \widetilde{\mathbf{m}} \times t/q \rceil$
  return m
end function

Algorithm 5.4 FV: Homomorphic operations

function FV.ADD($\mathbf{c}^a, \mathbf{c}^b$)
  $\mathbf{c}^+ \leftarrow (\mathbf{c_0}^a + \mathbf{c_0}^b, \mathbf{c_1}^a + \mathbf{c_1}^b)$
  return $\mathbf{c}^+$
end function
function FV.MULT($\mathbf{c}^a, \mathbf{c}^b$)
  $\widetilde{\mathbf{c}}_0 \leftarrow [\lfloor \mathbf{c_0}^a \times \mathbf{c_0}^b \times t/q \rceil]_q$
  $\widetilde{\mathbf{c}}_1 \leftarrow [\lfloor (\mathbf{c_0}^a \times \mathbf{c_1}^b + \mathbf{c_1}^a \times \mathbf{c_0}^b) \times t/q \rceil]_q$
  $\widetilde{\mathbf{c}}_2 \leftarrow [\lfloor \mathbf{c_1}^a \times \mathbf{c_1}^b \times t/q \rceil]_q$
  $\mathbf{c}^\times \leftarrow$ FV.RELIN($\widetilde{\mathbf{c}}_0, \widetilde{\mathbf{c}}_1, \widetilde{\mathbf{c}}_2, \gamma$)
  return $\mathbf{c}^\times$
end function
function FV.RELIN($\widetilde{\mathbf{c}}_0, \widetilde{\mathbf{c}}_1, \widetilde{\mathbf{c}}_2, \gamma$)
  $\mathbf{c}_0 \leftarrow [\widetilde{\mathbf{c}}_0 + \langle$ FV.WORDDECOMP$_{\omega,q}(\widetilde{\mathbf{c}}_2), \gamma_0 \rangle]_q$
  $\mathbf{c}_1 \leftarrow [\widetilde{\mathbf{c}}_1 + \langle$ FV.WORDDECOMP$_{\omega,q}(\widetilde{\mathbf{c}}_2), \gamma_1 \rangle]_q$
  $\mathbf{c} \leftarrow (\mathbf{c}_0, \mathbf{c}_1)$
  return c
end function

Algorithm 5.5 SHIELD: Utility functions

function SHIELD.BitDecomp($\mathbf{A} \in R_q^{N\times 2}$)
  $\mathbf{B} \in B_{R_q}^{N\times N}$
  for $i = 0$ to $N-1$ **do**
    for $j = 0$ to $\log_2 q - 1$ **do**
      $\mathbf{B}_{i,j} \leftarrow \mathbf{A}_{i,0}^{(j)}$
      $\mathbf{B}_{i,j+\log_2 q} \leftarrow \mathbf{A}_{i,1}^{(j)}$
    end for
  end for
  return B
end function
function SHIELD.BitDecompInv($\mathbf{B} \in B_{R_q}^{N\times N}$)
  $\mathbf{A} \in R_q^{N\times 2}$
  for $i = 0$ to $N-1$ **do**
    $\mathbf{A}_{i,0} \leftarrow \sum_{j=0}^{\log_2 q-1} 2^j \cdot \mathbf{B}_{i,j}$
    $\mathbf{A}_{i,1} \leftarrow \sum_{j=\log_2 q}^{N-1} 2^j \cdot \mathbf{B}_{i,j}$
  end for
  return A
end function

Algorithm 5.6 SHIELD: Key generation function

function SHIELD.GenKeys($\sigma_{key}, \sigma_{err}$)
  $\mathbf{t} \leftarrow D_{R_q,\sigma_{key}}$
  $\mathbf{a} \leftarrow U_{R_q}$
  $\mathbf{e} \leftarrow D_{R_q,\sigma_{err}}$
  $\mathbf{b} \leftarrow \mathbf{a}\cdot\mathbf{t} + \mathbf{e}$
  $\mathbf{P_k} = \begin{bmatrix}\mathbf{b} & \mathbf{a}\end{bmatrix}$
  $\mathbf{S_k} = \begin{bmatrix}\mathbf{1} \\ -\mathbf{t}\end{bmatrix}$
  return $(\mathbf{P_k}, \mathbf{S_k}) \in R_q^{1\times 2} \times R_q^{2\times 1}$
end function

Algorithm 5.7 SHIELD: Encryption/Decryption

function SHIELD.ENCRYPT($\mathbf{m}, \mathbf{P_k}, \sigma_{err}$)
  $\mathbf{r_{N\times 1}} \leftarrow B_{R_q}^{N\times 1}$
  $\mathbf{e_{N\times 2}} \leftarrow D_{R_q,\sigma_{err}}^{N\times 2}$
  $\mathbf{C_{N\times 2}} \leftarrow \mathbf{m} \cdot \mathrm{BDI}(\mathbf{I_N}) + \mathbf{r_{N\times 1}} \cdot \mathbf{P_k} + \mathbf{E_{N\times 2}}$
  return $\mathbf{C_{N\times 2}}$
end function
function SHIELD.DECRYPT($\mathbf{C}, \mathbf{S_k}$)
  $\mathbf{M} \leftarrow \mathbf{C} \cdot \mathbf{S_k}$
  $\mathbf{m} \leftarrow \lfloor 2/q \times \cdot \mathbf{M}_{0,0} \rceil$
  return m
end function

Algorithm 5.8 SHIELD: Homomorphic operations

function SHIELD.ADD($\mathbf{C}^a, \mathbf{C}^b$)
  $\mathbf{C}_+ = \mathbf{C}^a + \mathbf{C}^b$
  return $\mathbf{C}_+$
end function
function SHIELD.MULT($\mathbf{C}^a, \mathbf{C}^b$)
  $\mathbf{C}^\times = \mathrm{BITDECOMP}(\mathbf{C}^a) \times \mathbf{C}^b$
  return $\mathbf{C}^\times$
end function

On the security front, a unique common work is needed. At first we went in a similar fashion as Lepoint and Naehrig did in their comparative study of FV and YASHE that had been used several times by implementers.

Notation

We briefly introduce additional notation for the noise extraction. When $\mathbf{a} \leftarrow D_{R_q,\sigma_{key}}$ and $\mathbf{b} \leftarrow D_{R_q,\sigma_{err}}$, we note $\|\mathbf{a}\|_\infty = B_{key}$ and $\|\mathbf{b}\|_\infty = B_{err}$. B_0 refers to the upper bound of the noise for a fresh ciphertext, B_L denotes the noise bound after a multiplicative depth of L. We also introduce the expansion factor δ_R of a ring R, which bounds the product of two ring elements. The expansion can be expressed as

$$\delta_R = \sup_{\mathbf{a},\mathbf{b}\in R} \left\{ \frac{\|\mathbf{a}\cdot\mathbf{b}\|_\infty}{\|\mathbf{a}\|_\infty \|\mathbf{b}\|_\infty} \right\}$$

FV

The noise bound has been thoroughly studied in [LN14], thus we only recall some key aspects below.

Initial noise. To determine the initial noise, we apply the decryption procedure on a fresh ciphertext, focusing on the encryption of a 0:

$$\begin{aligned} \mathbf{c_0} + \mathbf{c_1 s} &= (-\mathbf{as} + \mathbf{e})\mathbf{u} + \mathbf{e_1} + (\mathbf{au} + \mathbf{e_2})\mathbf{s} \\ &= \mathbf{eu} + \mathbf{e_1} + \mathbf{e_2 s} \end{aligned}$$

Thus, the initial noise is $B_0 = B_{err}(1 + 2nB_{key})$.

Multiplicative noise. Following the approach in [LN14, Section 3.5] we can express an inequality that must be satisfied for decryption to be correct at depth L.

$$C_1^L B_0 + LC_1^{L-1} C_2 < (\lfloor q/t \rfloor - (q - \Delta t))/2$$

where

$$C_1 = \delta t(4 + \delta B_{key}), \quad C_2 = \delta^2 B_{key}(B_{key} + t^2) + \delta\omega l_{\omega,q} B_{err}$$

For binary messages it becomes:

$$C_1 = 2n(4 + nB_{key}), \quad C_2 = n^2 B_{key}(B_{key} + 4) + n\omega l_{\omega,q} B_{err}$$

SHIELD

The authors of [KGV16] only provided an asymptotic evaluation of SHIELD noise growth. Following the same approach as before, we develop below a more precise calculation, providing the constant terms.

Initial noise. To determine the initial noise, we apply the decryption procedure on a fresh ciphertext, focusing on the encryption of a 0:

$$\begin{aligned}\mathbf{C}\cdot\mathbf{S_k} &= (\mathbf{m}\cdot\mathrm{BDI}(\mathbf{I_N}) + \mathbf{r_{N\times 1}}\cdot\mathbf{P_k} + \mathbf{E_{N\times 2}})\cdot\mathbf{S_k}\\ &= \mathbf{r_{N\times 1}}\cdot\mathbf{P_k}\cdot\mathbf{S_k} + \mathbf{E_{N\times 2}}\cdot\mathbf{S_k}\end{aligned}$$

We set $\mathcal{E} = \mathbf{r_{N\times 1}}\cdot\mathbf{P_k}\cdot\mathbf{S_k} + \mathbf{E_{N\times 2}}\cdot\mathbf{S_{key}}$ and we have

$$\begin{aligned}\|\mathcal{E}[i]\|_\infty &\leq nB_{err} + B_{err} + n\cdot B_{err}\cdot B_{key}\\ &= B_{err}(1+n(1+B_{key}))\end{aligned}$$

Thus, the initial noise can be bounded by $B_0 = B_{err}(1+n(1+B_{key}))$.

Multiplicative noise. To determine the noise after a homomorphic multiplication in SHIELD, we apply the decryption procedure after the multiplication step. Recall that $\text{SHIELD.}\textsc{Mult}(\mathbf{C}^a,\mathbf{C}^b) = \mathrm{BD}(\mathbf{C}^a)\cdot\mathbf{C}^b$

$$\begin{aligned}\mathrm{BD}(\mathbf{C}^a)\cdot\mathbf{C}^b\cdot\mathbf{S_k} &= \mathrm{BD}(\mathbf{C}^a)(\mathbf{m}_b\mathrm{BDI}(\mathbf{I_N})\cdot\mathbf{S_k} + \mathcal{E}_b)\\ &= \mathbf{m}_b\cdot\mathrm{BD}(\mathbf{C}^a)\cdot\mathrm{BDI}(\mathbf{I_N})\cdot\mathbf{S_k} + \mathrm{BD}(\mathbf{C}^a)\cdot\mathcal{E}_b\\ &= \mathbf{m}_b\cdot\mathbf{C}^a\cdot\mathbf{S_k} + \mathrm{BD}(\mathbf{C}^a)\cdot\mathcal{E}_b\\ &= \mathbf{m}_a\cdot\mathbf{m}_b\cdot\mathrm{BDI}(\mathbf{I_N})\cdot\mathbf{S_k} + \mathbf{m}_b\cdot\mathcal{E}_a + \mathrm{BD}(\mathbf{C}^a)\cdot\mathcal{E}_b\end{aligned}$$

We set $\mathcal{E}_\times = \mathbf{m}_b\cdot\mathcal{E}_a + \mathrm{BD}(\mathbf{C}^a)\cdot\mathcal{E}_b$. To bound $\mathcal{E}_\times$, which is a vector, one must bound each elements. $\mathrm{BD}(\mathbf{C}^a)$ is always a $N\times N$-matrix of binary polynomials. Thus, each row of $\mathrm{BD}(\mathbf{C}^a)\cdot\mathcal{E}_b$ is a product/accumulation of $N = 2\log_2 q$ binary polynomials with polynomials bounded by $\|\mathcal{E}_b[i]\|_\infty$. After one homomorphic multiplication, the noise can be bounded by

$$\begin{aligned}\|\mathcal{E}_\times[i]\|_\infty &\leq \mathbf{m}_a\cdot B_0^a + 2n\cdot\log_2 q\cdot B_0^b\\ &\leq B_0(1+2n\cdot\log_2 q)\end{aligned} \tag{5.1}$$

Then, by an immediate induction, the noise after L homomorphic multiplications can be expressed as $B_L = B_0(1+2n\log_2 q)^L$. To be able to decrypt without error after L homomorphic multiplications, the final noise must be lower than $q/2$. We must have

$$q/2 > B_0(1+2n\log_2 q)^L$$

Better noise for multiplication. Unlike in FV, noise in SHIELD grows slowly if a ciphertext is multiplied by a fresh one. By carefully examining Equation 5.1, one can deduce that the noise of each ciphertext is independent. Thus, the multiplicative noise growth can be more finely managed. When a ciphertext is multiplied by L other fresh ciphertexts, the noise growth can be expressed as

$$B_L = B_0 + L(2n\log_2 q)B_0 = B_0(1+2Ln\log_2 q)$$

Table 5.1 – Maximum $\log_2 q$ for a given dimension n, where λ is the security level. $\sigma_{err} = 2\sqrt{n}$.

n	2048	4096	8192	16384
$\lambda = 80$ bits	89 bits	174 bits	348 bits	695 bits
$\lambda = 128$ bits	59 bits	114 bits	224 bits	444 bits

With batching. Earlier, we extracted noise parameters when $\mathbf{m} \in \{0, 1\}$. However, if one wants to use batch operations, the message is now a polynomial with coefficients in $\{0, 1\}$. This has a significant impact on the noise growth in SHIELD. We express the new bound of the noise $\mathcal{E}_\times$:

$$\begin{aligned}\|\mathcal{E}_\times[i]\|_\infty &\leq \|\mathbf{m}_b \cdot \mathcal{E}_a\|_\infty + \|\mathrm{BD}(\mathbf{C}^a) \cdot \mathcal{E}_b\|_\infty \\ &\leq n \cdot \|\mathbf{m}_b\|_\infty \|\mathcal{E}_a\|_\infty + 2n \cdot \log_2 q \cdot \|\mathcal{E}_b\|_\infty\end{aligned}$$

In the case of the optimised circuit for SHIELD, i.e. the second ciphertext is a fresh one, the noise new bound can be expressed as :

$$B_{i+1} = n \cdot B_i + 2n \cdot \log_2 q \cdot B_0$$

It is an arithmetico-geometric sequence of the form $B_{i+1} = a \cdot B_i + b$, where $a = n$ and $b = 2n \log_2 q B_0$. So $B_L = a^L(B_0 - r) + r$, with $r = \frac{b}{1-a}$.

5.2.4 Security

While the noise management determines the multiplicative depth and set a minimum q to keep computations correct, the security requirement asks for the opposite. It upper-bounds the size of the modulus for a given dimension n. This means that, sometimes, to ensure a given multiplicative depth one cannot only increase the modulus. The induced loss in security must be compensated by an increase in the dimension n. For a brief overview we put in Table 5.1 the maximal allowed $\log q$ for several dimensions at 80 and 128 bits of security.

As expected in cryptography, all the schemes presented here come with hardness results, provided by reductions to the Ring-LWE problem. This hard problem is one of the best candidates for post-quantum cryptography. There are no quantum attacks performing better than the classical ones. Yet, beyond these asymptotic reductions, we need concrete hardness results to choose the scheme parameters according to a security level objective, e.g. 80 bits or 128 bits. As we said above, one key component to make a pertinent study is to have all the schemes secure, and equally secure. Consequently, we focus our study on parameters that do not violate the security reductions. First and foremost is the error distribution, which should not be too small with respect to the dimension n [Reg05]. Both previous studies did not make this similar choice.

At first we tried to adapt and update the lattice reduction modelling as in [LN14] but we quickly shifted to the `lwe-estimator` from Albrecht [Albb]. We extensively discuss this matter in the previous chapters of the manuscript. The take-away conclusion is that the

estimator from Albrecht is being constantly updated with the publications about attack improvements and as of today could be used as a very good indicator for the security level of a given parameter set.

To produce the values for this study, we used it as is in `commit 61ac716`, setting our threshold at exactly 80 or 128 bits of security. For real applications with long term perspective, we advice to add some extra bits of margin. We already said it several times, huge efforts are conducted nowadays on cryptanalysis of post-quantum cryptography and new attacks can take away bits of security faster than Moore's Law.

5.3 Parameters in perspective

We automated the parameter derivation with scripts, under Matlab first. Then we moved to Sage to directly use Albrecht's estimator as a black-box. This allows to conveniently generate up-to-date values. In this section, we explore different settings: arbitrary circuit, optimised circuit, NWC, batching, and report concrete parameters for scheme comparison.

How to proceed? Real use-cases of homomorphic cryptography define requirements for the multiplicative depth L and a security level λ to achieve, then one needs to choose the corresponding security parameters. However we have three variables to adjust: the dimension n, the modulus q and the error size σ. Plus, a change to one of them could be compensated, amplified or cancelled by a change to another one, so we had to establish a sound procedure to avoid being caught in loops or achieving suboptimal parameters.

To remove one degree of freedom to the system, we decided to go for a fixed error size and set it to $\sigma = 2\sqrt{n}$. This choice is compliant with the hardness reduction and a bigger error would have a negative impact on the correctness. At least this is what is usually believed in, but we present in the next section how this is not always true, on average it seems a correct assumption.

So, the procedure goes as follows: for a tentative dimension n (start small), derive a lower bound for the modulus q to ensure correctness. Check if this lower bound is small enough so that with this dimension n (and σ), the security is good enough. If it is, the triplet (n, q, σ) satisfies both correctness and security requirements; if not, try again to find a modulus, this time with a bigger n. We formalize this procedure in Algorithm 5.9. The procedure MinModulus involves the noise growth equations established previously and SecurityLevel is a call of Albrecht's estimator.

5.3.1 Multiplicative depth for an arbitrary binary circuit

Table 5.2 provides parameters for FV and SHIELD for 80 and 128 bits of security. They are extracted in the proven-hardness regime, that is to say $\sigma_{err} = 2\sqrt{n}$ for each scheme.

Values for SHIELD seem the best in the tables. However the number of sub-polynomials for a given ciphertext explodes because it is proportional to $\log_2 q$ for SHIELD. For example, with $L = 5$, a ciphertext in SHIELD contains $2 \times N = 4 \times \log_2 q = 480$ polynomials of degree-2772 with 120 bits coefficients, whereas in FV a ciphertext is only two polynomials

Algorithm 5.9 Determine $(n, \sigma$ and $q)$ parameters from (L, λ) for a given scheme

```
function ChooseParam(scheme, L, λ)
    n ← 0
    repeat
        n ← n + 1
        σ ← 2√n
        q ← MinModulus(n, L, scheme)
    until SecurityLevel(n, σ, q) > λ
    return n, σ, q
end function
```

Table 5.2 – Parameters for FV and SHIELD, where λ is the security level and L the multiplicative depth. Arbitrary circuit.

(a) Selection of parameters for FV. Binary key, $\sigma_{err} = 2\sqrt{n}$.

L	$\lambda = 80$ bits				$\lambda = 128$ bits			
	$\omega = 32$ bits		$\omega = 64$ bits		$\omega = 32$ bits		$\omega = 64$ bits	
	$\log_2 q$	n	$\log_2 q$	n	$\log_2 q$	n	$\log_2 q$	n
1	54	1188	87	1982	55	1878	88	3106
5	159	3711	193	4507	166	6014	200	7292
10	303	7120	337	7917	317	11625	351	12898
15	454	10715	489	11549	475	17507	509	18729
20	611	14405	645	15187	639	23491	673	24755

(b) Selection of parameters for SHIELD. Binary key, $\sigma_{err} = 2\sqrt{n}$.

L	$\lambda = 80$ bits		$\lambda = 128$ bits	
	$\log_2 q$	n	$\log_2 q$	n
1	36	752	38	1247
5	120	2772	124	4454
10	238	5597	246	9005
15	364	8556	376	13834
20	495	11685	511	18838

of degree-3711 with 159 bits coefficients. Consequently, in the case of an arbitrary binary circuit, FV is best.

5.3.2 Multiplicative depth for an optimised circuit

As stated in the previous section, SHIELD seems very costly for arbitrary circuits. However, all third generation schemes have a really interesting feature: when a ciphertext is multiplied by a fresh ciphertext, the noise growth is additive instead of multiplicative for binary messages. Table 5.3 provides parameters for SHIELD for such optimised circuit. FV is omitted here, because it presents no particular optimisation.

Table 5.3 – Parameters for SHIELD, where λ is the security level and L the multiplicative degree. Optimized circuit. Binary message (No batching). Binary key, $\sigma_{err} = 2\sqrt{n}$.

L	$\lambda = 80$ bits		$\lambda = 128$ bits	
	$\log_2 q$	n	$\log_2 q$	n
1	36	752	38	1247
5	38	803	40	1323
10	40	849	41	1363
15	40	854	42	1396
20	41	869	43	1429

Results are very impressive, SHIELD scales to large multiplicative degree with nearly no impact on n and q. For 80 bits of security, the modulus only increases by 5 bits between a multiplicative depth of 1 and 20 when the degree of the associated cyclotomic polynomial remains under 1024. As a reminder from Table 5.2, FV requires at least $n = 14405$ and $\log_2 q = 611$ bits for a multiplicative depth of 20.

SHIELD is clearly better than FV in this setting, which is not about evaluating circuit of depth L for *all* inputs, yet still a degree-L function.

5.3.3 The case of the Negative Wrapped Convolution

As we saw in the tables so far, implementing homomorphic schemes requires to handle polynomials of very high degree with large coefficients. A challenge that arises is that of polynomial multiplication.

More on polynomial multiplication techniques

Recall that there are three families of algorithms for that:

- The schoolbook algorithm which has an asymptotic complexity of $\mathcal{O}(n^2)$
- More advanced algorithms like Karatsuba-Ofman [KO63] or Toom-Cook [Too63] which have asymptotic complexities of $\mathcal{O}(n^{1.58})$ and $\mathcal{O}(n^{1.465})$ respectively
- And *Fast Fourier Transform (FFT)* algorithms with the best asymptotic complexity of $O(n \log n)$.

The final choice on the polynomial multiplication algorithm will greatly depend on the size of operands and on the constraints from the evaluating environment (software/hardware).

For polynomials with integer modular arithmetic, FFT can be replaced by a specific *Number Theoretic Transform (NTT)* algorithm. This is a specialisation of the FFT in finite fields, implying only modular integer arithmetic with a prime modulus q instead of complex numbers in FFT. We recall the formal definition of NTT:

Let N be a power of 2, q a prime modulus such as $q \equiv 1 \mod 2N$, and w be a primitive N^{th} root of unity in $\mathbb{Z}_q$. The NTT/iNTT of a given polynomial $\mathbf{a} = (a_0, \cdots, a_{N-1})$ is defined

by:

$$\mathrm{NTT}(\mathbf{a}) = \left(\sum_{j=0}^{N-1} a_j w^{ij} \right)_{0 \le i \le N-1}$$
$$\mathrm{iNTT}(\hat{\mathbf{a}}) = \left(n^{-1} \sum_{j=0}^{N-1} \hat{a}_j w^{-ij} \right)_{0 \le i \le N-1}$$

For two polynomials $\mathbf{a}$ and $\mathbf{b}$, the polynomial multiplication can be computed as follows:

$$\mathrm{iNTT}(\mathrm{NTT}(\mathbf{a}) \odot \mathrm{NTT}(\mathbf{b})) = \mathbf{a} \cdot \mathbf{b} \mod X^N - 1$$

where $\odot$ denotes the component-wise multiplication of two vectors. This version NTT is called *Positive Wrapped Convolution* (PWC) and is not perfectly suited for homomorphic encryption because $X^N - 1$ is not a cyclotomic polynomial. Indeed, we do not want the polynomial reduction by $X^N - 1$, so we must double the size of the NTT and then perform the polynomial modular reduction manually.

Yet, with a minor modification, the NTT can be turned a *Negative Wrapped Convolution* (NWC) which computes $\mathbf{a} \cdot \mathbf{b} \mod X^N + 1$. Then, because $X^N + 1$ is a cyclotomic polynomial, the polynomial modular reduction is directly integrated during NTT computations, at no extra cost.

However, the NWC technique is incompatible with batching as we shall see below. When factoring $X^n + 1$ in $\mathbb{Z}_2[X]$, the resulting polynomial is $(X + 1)^N$, which has a unique factor, namely $(X + 1)$, so it is not possible to "pack" several messages. This is incompatible with the batching technique presented in Section 5.3.4. Thus, for binary messages, the NWC, which is optimised for performance, is not well suited for parallel computations. For non binary messages, it would be possible to find some configurations compatible with batching by selecting a particular modulus prime q.

Parameters under NTT NWC constraints

We provide in Table 5.4 the parameters for FV and SHIELD under the constraints from NWC. Parameters are selected to maximize the multiplicative depth for a given n, which is necessarily a power of 2, because the NWC NTT set the cyclotomic polynomial to $X^n + 1$. When compared to the previous case, this slightly increases the size of the modulus, for a given multiplicative depth.

For example with FV, for a multiplicative depth of 4, optimised parameters are $n = 3065$ and $\log_2 q = 132$. In a NWC NTT scenario, new parameters are $n = 4096$ and $\log_2 q = 135$ bits. Thus, the ciphertexts are slightly larger when compared to optimised ones, but the computation time will be better than with a NTT multiplication of size $2N$.

For SHIELD, parameters seem quite independent of the multiplicative depth. Because the polynomial degree is oversized due to NWC, a security of $\lambda = 80$ bits requires $n = 1024$, cf Table 5.3, and we can then go to very high depth. Similarly, $n = 2048$ is required for $\lambda = 128$ bits.

Table 5.4 – Parameters for FV and SHIELD in the case of the NWC NTT, where λ is the security level and L the multiplicative depth. Binary key, $\sigma_{err} = 2\sqrt{n}$. Reminder: no batching with the NWC NTT.

(a) Parameters for FV.

n	$\lambda = 80$ bits				$\lambda = 128$ bits			
	$\omega = 32$ bits		$\omega = 64$ bits		$\omega = 32$ bits		$\omega = 64$ bits	
	$\log_2 q$	L	$\log_2 q$	L	$\log_2 q$	L	$\log_2 q$	L
2048	79	2	87	1	55	1	×	×
4096	159	5	166	4	109	3	88	1
8192	333	11	337	10	195	6	200	5
16384	675	22	677	21	443	14	414	12

(b) Parameters for SHIELD.

$\lambda = 80$ bits, n=1024		$\lambda = 128$ bits, n=2048	
$\log_2 q$	L	$\log_2 q$	L
36	1	38	1
38	5	40	5
40	10	41	10
40	15	42	15
41	20	43	20

5.3.4 Parameters for batching

How does it works?

For both FV and SHIELD, the plaintext is a polynomial in R_t for some integer $t \geq 2$. Because of the expansion of homomorphic encryption, we would like to have integer messages, i.e. $t >> 2$. However, this integer representation prevents us from doing interesting homomorphic operations. Indeed, all algorithms doing comparison require dealing with binary messages. This latter representation also has some drawbacks. First, to perform an integer addition or multiplication on an integer, one must reconstruct the binary circuit of these operators. Second, the size of ciphertexts is strongly impacted because a ciphertext holds only one bit of plaintext, unlike with integer representation where tens of plaintext bits get encrypted into one ciphertext. Yet, recent research [Ang17] demonstrates that using a binary encoding for plaintext data is optimal.

To control the ciphertext expansion issue, the *batching* technique was introduced [SV14]. This trick allows to "pack" several messages into one single ciphertext. To do so, the associated cyclotomic polynomial must be reducible in $\mathbb{Z}_2[X]$. Then, a polynomial CRT is applied to pack the messages, with one message per factor.

New parameters for SHIELD

As stated above, the batching technique is very useful to reduce the ciphertext expansion. Table 5.5 provides parameters for SHIELD when the batching technique is used, in an

Table 5.5 – Parameters of SHIELD for 80 bits of security when batching is enabled, where λ is the security level and L the multiplicative depth. Binary key, $\sigma_{err} = 2\sqrt{n}$.

L	$\log_2 q$	n
1	36	752
2	47	1016
3	59	1306
4	71	1596
5	84	1911
10	149	3476

optimised circuit as described in Section 5.3.2. FV is not represented because batching does not imply new noise growth equations as SHIELD. For FV, we detail below the precise choices for n depending on the number of slots.

Unlike when the messages are binary, SHIELD parameters becomes sensitive to the multiplicative depth. As early as depth 3, the dimension goes over 1024 and implies an associated NTT of size 2048. Moreover, the modulus q grows significantly with the depth, on average 12 more bits per level which leads to more and more polynomials for a given ciphertext. For a multiplicative depth of 10, SHIELD with batching requires 596 polynomials of degree 3476 with coefficients of 149 bits, while without batching it only requires 160 polynomials of degree 849 with coefficients of 40 bits. The use of batching with SHIELD is not recommended.

Focus on polynomials choice for batching with FV

For completeness we keep this section though it is based on work from Vincent Migliore only.

We have investigated the structure and the repartition of cyclotomic polynomials in order to measure the practicality of batching. Because there is no known efficient NTT to perform polynomial modular reduction with arbitrary cyclotomic polynomial (apart from the cases of NWC with $X^n + 1$ as presented in Section 5.3.3), the polynomial modular reduction must be implemented manually. Thus, batching can only be practical if we have a cyclotomic polynomial that allows efficient reduction.

Because the polynomial modular reduction complexity is closely related to the Hamming weight of the cyclotomic polynomial (*i.e.* the number of non-zero monomials), we have minimised as much as possible this parameter. Table 5.6 provides the conclusions of our exploration. We only have investigated batching for FV, since SHIELD does not seem practical for batching as discussed above. For each multiplicative depth, we have extracted the four cyclotomic polynomials with the lowest Hamming weight and compatible with batching. As can be seen, batching can be implemented for each multiplicative depth with various number of batches. In addition, for all parameters, the Hamming weight is very small when compared to the degree of the cyclotomic polynomial, further supporting the practicality of batching.

Table 5.6 – Parameters of FV for 80 bits of security when batching is enabled, where L is the multiplicative depth, batching the number of packed operations, m the rank of the cyclotomic polynomial and weight of the number of non-zero monomials in the associated cyclotomic polynomial. Binary key, $\sigma_{err} = 2\sqrt{n}$

Range of n	L	batching	weight	m	Actual n
$[1024, 2048]$	1	2	7	3375	1800
		6	9	3087	1764
		12	33	2835	1296
		24	59	2925	1440
	2	2	7	3645	1944
		6	17	3159	1944
		18	49	2997	1944
		20	57	4125	2000
$[2048, 4096]$	3	2	7	5625	3000
		6	9	5103	2916
		18	25	4617	2916
		30	49	3875	3000
	4	2	7	6075	3240
		6	23	4459	3528
		12	33	7875	3600
		20	57	7425	3600
	5	2	17	6591	4056
		12	33	8505	3888
		24	59	7605	3744
		40	65	5125	4000
$[4096, 8192]$	6	2	7	9375	5000
		10	17	6875	5000
		12	33	11025	5040
		20	57	9075	4400
	7	2	7	10125	5400
		6	9	9261	5292
		18	25	9747	6156
		20	57	12375	6000

5.3.5 Keys and ciphertexts sizes

One of the key aspect for the practicality of homomorphic encryption is obviously the size of keys and ciphertexts. In order to fairly evaluate FV and SHIELD, we have compared the volume of data required for each scheme in a scenario requiring 8 bits of information. Figure 5.2 provides the conclusions of the study. For FV, the size of relinearisation keys are also included because they are required during the homomorphic multiplication.

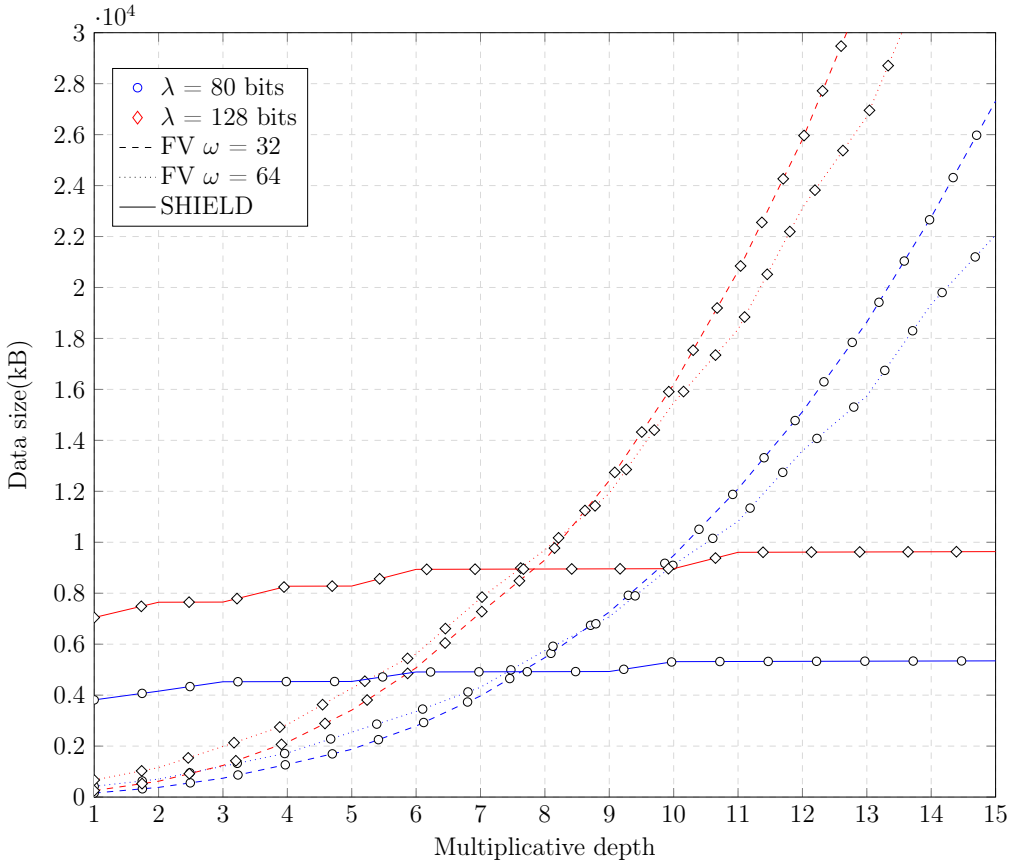
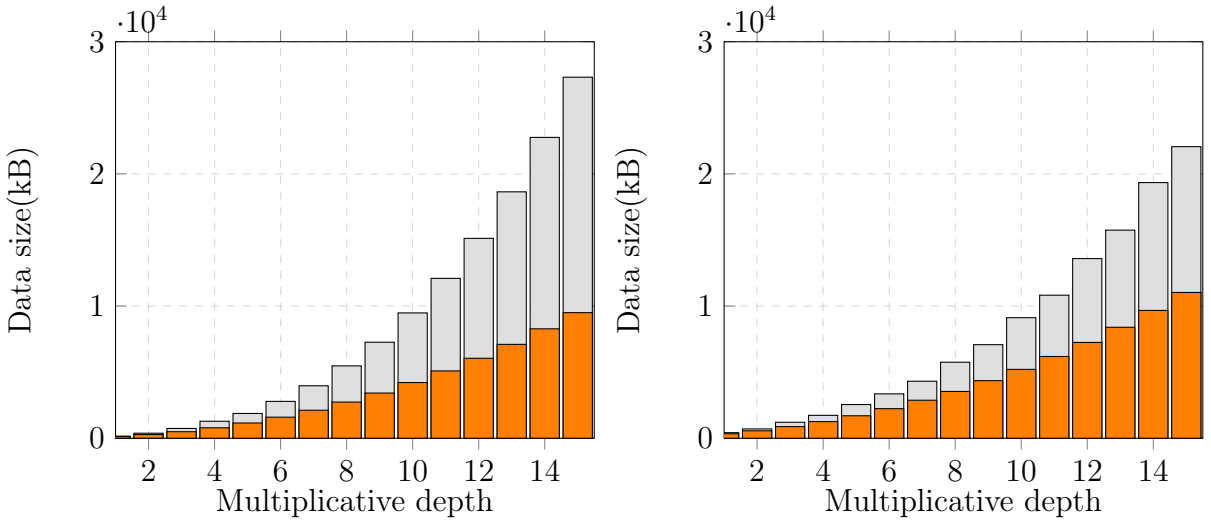


Figure 5.2 – Data size required for FV and SHIELD in a scenario with 8 encryptions.

For small multiplicative depths, namely under 8, FV requires a lower amount of data than SHIELD. This is because SHIELD requires large parameters as of the first multiplicative depths. But for larger depths, the improved noise management of SHIELD is highly beneficial. The main issue for FV is the size of the relinearisation key. For a multiplicative depth of 15, it is as large as 17.8 MB, when SHIELD does not require such a key. It can be reduced a bit by enlarging ω at an additional computation cost.

We deepen our analysis on the size of the relinearisation key. In Fig. 5.3a and 5.3b, we show for FV the respective sizes of this additional key and 8 ciphertexts. Both are transmitted from client to server, and we can see that going from $\omega = 32$ to $\omega = 64$ significantly decreases the relinearisation key sizes, and with it, the transmission overhead. We shall also note that when the server sends a computation result back to the client, it does not need to include this extra key material, and in this case only the orange bars corresponding to ciphertexts are of interest. Hence, these figures illustrate both the different transmission overhead for client-server and server-client communications, and the interest of using $\omega = 64$ instead of $\omega = 32$ for the client-server upload.



(a) $\omega = 32$ (b) $\omega = 64$

Figure 5.3 – Histograms showing the respective size of ciphertexts and keys for FV with $\lambda = 80$. The lower (orange) part is the cumulative size of 8 ciphertexts and the upper part is the relinearisation key.

5.4 Smallest error is not always the best

As we described in the previous section, security and correctness directly depend on all the parameters: dimension, modulus and error size. Few relations exist and are not necessarily written black on white in the literature so we take the chance here to highlight them, before pushing the reasoning a bit further.

For a scheme to be correct, the error shall not be too big in front of the modulus, i.e. $\sigma << q$. We can grasp the idea that big errors will induce wrap-around q and mess things up. Yet for security, the error shall not be to small, $\sigma > \omega(\sqrt{n})$, otherwise the "masking" action we intend with error addition is not effective. So, we can say that

- Increasing q, increase correctness but reduce security.
- Increasing σ, decrease correctness but increase security.

Also, increasing n usually increase the difficult of the underlying problems so increase security, yet it magnifies the effect of the errors so can decrease the correctness.

These three trends and their respective weights made us derive our Algorithm 5.9 to draw parameters, with a fixed $\sigma = 2\sqrt{n}$. We said we do not want to take it smaller, because of the hardness reduction (see [Pei16] for extensive details). Yet we might try to make it bigger with the hope to gain security without breaking correctness and so it may result in smaller parameters. We investigated this behaviour in more detail after we noticed it accidentally.

We took the case of FV for a depth 10 and 80 bits of security and derive parameters as before while fixing $\sigma = x\sqrt{n}$ with x varying from 2 to 200. Figure 5.4 displays the size of

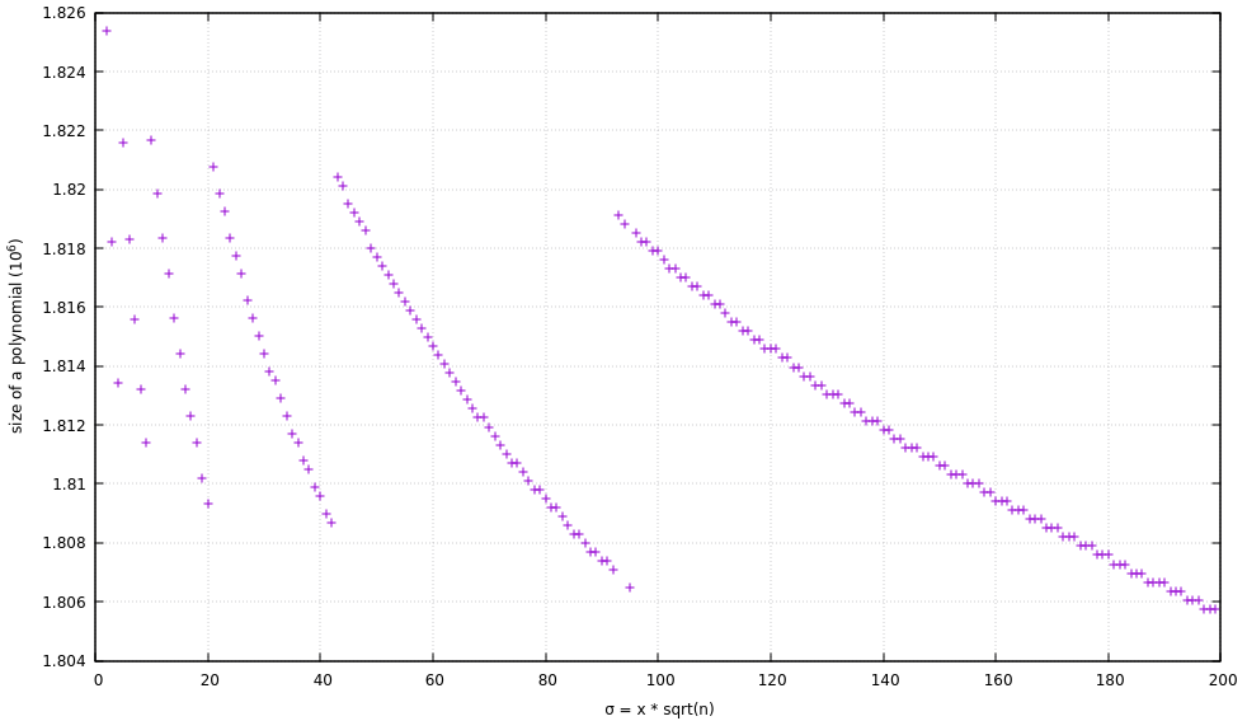


Figure 5.4 – Ciphertexts and key sizes in term of σ

the resulting polynomials of degree n with coefficient size $\log q$ (again using Alg. 5.9. The ciphertext sizes and key sizes are multiple of n and $\log q$.

Let us explain the phenomenon. We can see piecewise decreasing curves. For each of these pieces, increasing sigma makes the polynomials smaller. Indeed, a bigger error brings more security and so a smaller dimension is enough to reach the target security level. This is the behaviour we wished for. Yet, we observe a threshold effect of the size. The reason is that, if we continue to increase the error, the correctness is diminishing to the point when it no longer works. In such cases, we have to take a bigger modulus q (one more bit) to restore correctness, hence the jumps of the curve.

Taking one step back from the values, we see that the variation is limited, roughly 1% on this example. Yet every optimisation is interesting when considering deployment. However, in our exploration space of optimal parameters with three degree of freedom (n, σ, q), we had hoped that this third requirement (that of minimising the ciphertext) would have given us a global optimum, but it does not seem to be the case.

5.5 Implementation performance comparison

Now that we have done a thorough analysis of the parameters at hand within the different schemes, it would be of great interest to compare the *actual* performance of the scheme. It was a goal for us to continue in this direction. However comparing third party libraries in

different languages or needing extensive tuning is always questionable.

In [LN14], they chose to implement FV and YASHE' to compare them. This is a good approach because the skill of the programmer and the optimisation level would be very similar for all scheme. But it is quite time-consuming and there is always some more optimisation that can be done. Another approach (as in [HF17]) involves using third party libraries with their default settings. This is also imperfect because some tuning could be done here and there to make things better. This would require to have good insights on all libraries.

To circumvent these limits, our idea was to take NFLlib [Qua], upon which there is a FV implementation [Cry] and write a similar implementation of SHIELD, with the aim to keep both implementations very similar. It would benefit from a highly optimised library for the core (costly) operations, while also keeping the scheme implementation comparable because they consists only in a "thin" layer over NFLlib. This work is partially done but not good enough to have been released yet [BM]. And so, the performance study is left for near future work.

Conclusion

We have detailed in this chapter our comparative study of homomorphic encryption schemes. This work was accepted for a special issue of *IEEE Transactions on computers* on *Cryptographic Engineering in a Post-Quantum World* [**?**]. Giving clues to an industrial implementer is the main outcome of our work.

Throughout this in-depth analysis, we were able to draw some general conclusions and pinpoint the most dimensioning questions to consider when going for homomorphic encryption use. Our study covered the most promising schemes as of 2016 and in the next chapter we will move on to the latest generation of schemes.

This work does not set a final point on the matter. The latest proposals discussed below extensively change the criteria for choosing an encryption scheme. For an application we will always need to pick a scheme, and also maybe one of its implementations if several exist. In the future we may have heuristics for automatic selection of scheme and implementation. As of now, we need them to mature more. The effort for a common API [1] is a very good step forward.

[1] `https://github.com/bristolcrypto/HEAT`

Chapter 6

Construction of a new scheme

Becoming the chased mouse.

As our last contribution, we endorsed a designer role. Thanks to the invitation of Léo Ducas, I joined the Centrum Wiskunde & Informatica (CWI) in Amsterdam between March and May 2017. During my stay, I worked with him and Max Fillinger on a new scheme that we called HE8, for "Homomorphic Evaluation over 8 bits". This new scheme is an improvement of FHEW and belongs to the new trend of *bootstrapped schemes*. It was accepted for Africacrypt 2018 [**?**]. The roadmap for this chapter is as follows: first we shall review the proposals of bootstrapped schemes, the fourth generation of homomorphic schemes. Then we

describe the homomorphic gate, its elementary procedures and optimisations. And finally, we focus on the implementation aspects that I took care of during my time at CWI.

6.1 Introduction to the fourth generation

In the beginning of this dissertation we reviewed the young history of fully homomorphic encryption from Gentry's proposal in 2009 to the most mature BGV or FV as of this time. Yet a new trend of homomorphic schemes arises, that of bootstrapped schemes. We could refer to them as the fourth generation but to the best of our knowledge, no one did so before. For a refresher on the different generations, consult Section 2.4.1. This new kind of scheme has the interesting feature that the elementary operations supported by the scheme show no error growth. This trend emerged in 2014 with the work of Alperin-Sheriff and Peikert [AP14]. Ducas and Micciancio released the scheme FHEW [DM15] that can evaluate a NAND gate in time less than 1 second. More recently Chillotti et al. [CGGI16] introduced TFHE which achieve even better performances: less than 0.1 second for common gates (NOT,AND, NAND, OR, XOR).

Our work builds upon the ideas of FHEW and introduces orthogonal improvements to that of [CGGI16] with the aim of building an 8 bits generic gate. Concurrent work from the TFHE team [CGGI17] brings even more improvements to TFHE. They released their work at the end of April 2017, exactly when we finished our implementation of HE8.

6.2 Additional preliminaries

6.2.1 Rings

Our FHE scheme uses *circulant convolution rings* (or, for short, *circulant rings*). Circulant rings of degree d will be denoted with indeterminate T: $R_d = \mathbb{Z}[T]/(T^d - 1)$. We fix two distinct odd primes p and q. When speaking specifically of rings R_p, R_q, and R_{pq} we shall use indeterminates X, Y and Z, respectively. We write $\tilde{R}_d$ for the cyclotomic ring $\mathbb{Z}[\tilde{T}]/\Phi_d(\tilde{T})$ where $\Phi_d(\tilde{T})$ is the d-th cyclotomic polynomial.

We define the (normalised) *trace* function[1] as follows: we let $\mathrm{Tr}^*_{R_d/\mathbb{Z}} : R_d \to \mathbb{Z}, a \mapsto a_0$. If d is clear from context, we simply write this function as Tr^*. We let $\mathrm{Tr}^*_{R_{pq}/R_p} : R_{pq} \to R_p$ be the linear function defined by

$$\mathrm{Tr}^*_{R_{pq}/R_p}(Z^k) = \begin{cases} X^{k/q} & \text{if } q|k \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that: $\mathrm{Tr}^*_{R_d/\mathbb{Z}}$ and $\mathrm{Tr}^*_{R_{pq}/R_p}$ are linear, and: $\mathrm{Tr}^*_{R_p/\mathbb{Z}} \circ \mathrm{Tr}^*_{R_{pq}/R_p} = \mathrm{Tr}^*_{R_{pq}/\mathbb{Z}}$.

[1] This is simply a special case of the usual definition of the trace function, but we do not need the general definition here.

6.2.2 Circulant LWE and reduction to Ring-LWE

It is well known that the naive decisional version of Ring-LWE is insecure over circulant rings, simply by exploiting the CRT decomposition. Say that d is prime, and note that $R_d/QR_d \simeq \tilde{R}_d/Q\tilde{R}_d \times \mathbb{Z}/Q\mathbb{Z}$ if Q is co-prime to d, so one may mount an attack on the $\mathbb{Z}/Q\mathbb{Z}$ part (projecting to this part corresponds to evaluating the polynomial at 1, and therefore maintain smallness of the error). However, this does not mean that such rings are inherently insecure: the NTRU cryptosystems [HPS98, HHGP+03] use circulant rings, choosing the secret key and errors that evaluate to a fixed known value (say 0) at 1.

This suggests a strategy to construct a variant of Ring-LWE over circulant rings that would be as secure as the cyclotomic Ring-LWE, simply by lifting all elements $\tilde{x} \in \tilde{R}_d/Q\tilde{R}_d$ to $x \simeq (\tilde{x}, 0)$, yet this reverse CRT operation may not keep small elements small. Léo Ducas showed how to circumvent this obstacle in the published version of this work [BDF17]. We detail the practical error sampling procedure in Section 6.2.4.

6.2.3 LWE encryption

We recall the definition of the most basic LWE symmetric encryption scheme (see [BFKL94, Reg05, ACPS09]). LWE symmetric encryption is parametrised by a dimension n, a plaintext modulus $t \geq 2$, a ciphertext modulus $Q = n^{O(1)}$ and an error distribution χ. The message space of the scheme is $\mathbb{Z}_t$. (Typically, $e \leftarrow \chi$ satisfies the condition $|e| < Q/2t$, and $t = 2$ is used to encrypt message bits.) The (secret) key of the encryption scheme is a vector $\mathbf{s} \in \mathbb{Z}_Q^n$, which may be chosen uniformly at random, or as a random short vector. The encryption of a message $m \in \mathbb{Z}_t$ under key $\mathbf{s} \in \mathbb{Z}_Q^n$ is

$$\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + \lfloor Q/t \rceil \, m \bmod Q) \in \mathbb{Z}_Q^{n+1}$$

where $\mathbf{a} \leftarrow \mathbb{Z}_Q^n$ is chosen uniformly at random. A ciphertext $(\mathbf{a}, b)$ is decrypted by computing

$$m' = \lfloor t(b - \langle \mathbf{a}, \mathbf{s} \rangle)/Q \rceil \bmod t \in \mathbb{Z}_t.$$

We write $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}^{t|Q}(m)$ to denote that $\mathbf{c}$ is an LWE-encryption of m, and $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}^{t|Q}(m; E)$ if $\mathbf{c}$ is a random LWE-ciphertext such that $\mathbf{c} = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \lfloor Q/t \rceil \, m + e)$ where e is a subgaussian random variable with parameter E. The error of $\mathbf{c} = (\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{t|Q}(m)$ is $\mathsf{err}(\mathbf{c}) = (b - \langle \mathbf{a}, \mathbf{s} \rangle - \lfloor Q/t \rceil \, m) \bmod Q$, reduced modulo Q to the centered interval $[-Q/2, Q/2)$. Notice that the error $\mathsf{err}(\mathbf{a}, b)$ depends not just on $(\mathbf{a}, b)$, but also on $\mathbf{s}, Q, t$ and m.

We use this LWE encryption to construct two encryption schemes, Circulant-LWE and Circulant-GSW (based on [GSW13]), which we need for our *homomorphic accumulator* (see Section 6.3). We do not specify any decryption procedures since we do not need any for the homomorphic accumulator.

Circulant-LWE encryption scheme

We let R, $\tilde{R}$, d, t and Q be as in Sections 6.2.1 and 6.2.3. The Circulant-LWE scheme over R consists of the following algorithms:

- KeyGen: Output a uniformly random element s of $\tilde{R}$.
- $\mathsf{Enc}_s(m)$ for $m \in R/tR$: Let (a, b) be a sample from the Circulant-LWE distribution over R with secret s and output $(a, b' = b + \lfloor Q/t \rceil \cdot m)$.

We also define an n-dimensional variant of the scheme where the key is $\mathbf{s} \in R^n$, $\mathbf{a}$ is a random vector in R^n and the product $a \cdot s$ is replaced by the inner product over R: $\langle \mathbf{a}, \mathbf{s} \rangle = \sum_{i=1}^{n} \mathbf{a}_i \cdot \mathbf{s}_i$.

Thanks to the reduction of Circulant-LWE, we have that the Circulant-LWE scheme is CPA-secure for messages of the form $m = X^k$ if the decisional $\tilde{R}$-LWE problem is hard.

We write $\mathbf{c} \in R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m; E)$ if $\mathbf{c} = (\mathbf{a}, \mathbf{a}^t\mathbf{s} + \lfloor \frac{Q}{t} \rceil m + \mathbf{e})$ for some random error vector $\mathbf{e}$ that is E-subgaussian. We extend the notation to $\mathbf{C} \in R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(\mathbf{m}^T)$ for message $\mathbf{m} \in R_t^k$ that are vectors, meaning that the i-th column $\mathbf{C}_i$ of $\mathbf{C}$ is in $R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m_i)$. Furthermore, we write $\mathsf{err}(\mathbf{c})$ for the error term e in $\mathbf{c}$. We may drop the information on the error size where it is not needed to make the notation easier.

Circulant-GSW encryption scheme

We let R, $\tilde{R}$, d, t and Q be as in Sections 6.2.1 and 6.2.3, and $\mathbf{G}$ as in Section 2.1. Furthermore, let $t \geq 2$ be the plaintext modulus and B an integer ≥ 2, let K be the smallest integer such that $B^K \geq Q$.

The Circulant-GSW scheme is described by the following algorithms:

- KeyGen: Sample a uniformly random s from $\tilde{R}$.
- $\mathsf{Enc}_s(m)$ for $m \in R/tR$: Generate a matrix $\mathbf{A} \in R^{2K \times 2}$ where each row is a sample from the Circulant-LWE distribution with secret s. Output $\mathbf{A} + \lfloor Q/t \rceil \cdot m\mathbf{G}$.

We also define a n-dimensional variant of the scheme where $\mathbf{A} \in R^{(n+1)K \times (n+1)}$ and whose rows are samples from the n-dimensional Circulant-LWE and where $\mathbf{G}_1$ is replaced by $\mathbf{G}_n$.

As before, we have that the Circulant-GSW scheme is CPA-secure if the decisional $\tilde{R}$-LWE problem is hard.

We write $\mathbf{C} \in R_d\mathsf{GSW}_s^{t|Q}(m, E)$ if $\mathbf{C} = (\mathbf{a}, \mathbf{a}s + \mathbf{e}) + \lfloor \frac{Q}{t} \rceil \cdot m\mathbf{G}$, and the components of $\mathbf{e}$ are independent E-subgaussian variables. We write $\mathsf{err}(\mathbf{C})$ for the error vector $\mathbf{e}$ in $\mathbf{C}$. We may drop the information on the error size where it is not needed, to make the notation easier.

6.2.4 Simpler error distribution in CLWE for practice

In practice, most FHE schemes do not follow precisely the Ring-LWE problem definition admitting reduction to worst-case problem [LPR10]. For example, HElib [Hal, HS15] uses Ring-LWE with spherical errors in the coefficient embedding, and very sparse ternary secrets, ignoring the co-different ideal $R^\vee$. The TFHE scheme [CGGI16] also relies on Ring-LWE with ternary secrets, which is known to reduce to the regular Ring-LWE. Cutting such corners appears quite crucial to error growth management and therefore efficiency. We will follow this approach, and adjust the distributions as follows.

- First we sample secrets and error isotropically. Respecting the symmetries seems a must in the light of recent analysis [CIV16, Pei16].
- Then we choose to use ternary secrets $\mathbf{s}$, which, as in previous schemes, leads to serious performance improvements due to smaller error growth. It has recently been shown that such choices make lattice attacks somewhat faster [Alb17], especially when $\mathbf{s}$ is very sparse, so we will account for this refined analysis when measuring the concrete security of our proposed parameters.

Sampling of a. We sample $\mathbf{a}$ uniformly in $R_d/(QR_d)$ under the constraint that $\mathbf{a}(1) = 0 \mod Q$. We achieve this by choosing all the coefficients a_i at random for $i \geq 1$, and setting $a_0 = -\sum_{i>0} a_i \mod Q$.

Sampling of s. When d is prime, we sample a ternary $\mathbf{s}$ of density $\delta = 2/3$ by choosing exactly $\lfloor \delta d/2 \rfloor$ coefficients set to 1 and $\lfloor \delta d/2 \rfloor$ coefficients set to -1. This implies that $\mathbf{s}(1) = 0$, and $\|\mathbf{s}\|^2 = 2\lfloor \delta d/2 \rfloor$. Indeed, we find it preferable to fix its length to avoid sampling sparse keys that would be substantially weaker.

Sampling of e. We wish to sample errors $\mathbf{e}$ with variance σ in a way that ensures $\mathbf{e}(1) = 0$. We set:

$$\mathbf{e} = \sum_{i=0}^{\sigma^2 d/2} T^{a_i} - T^{b_i},$$

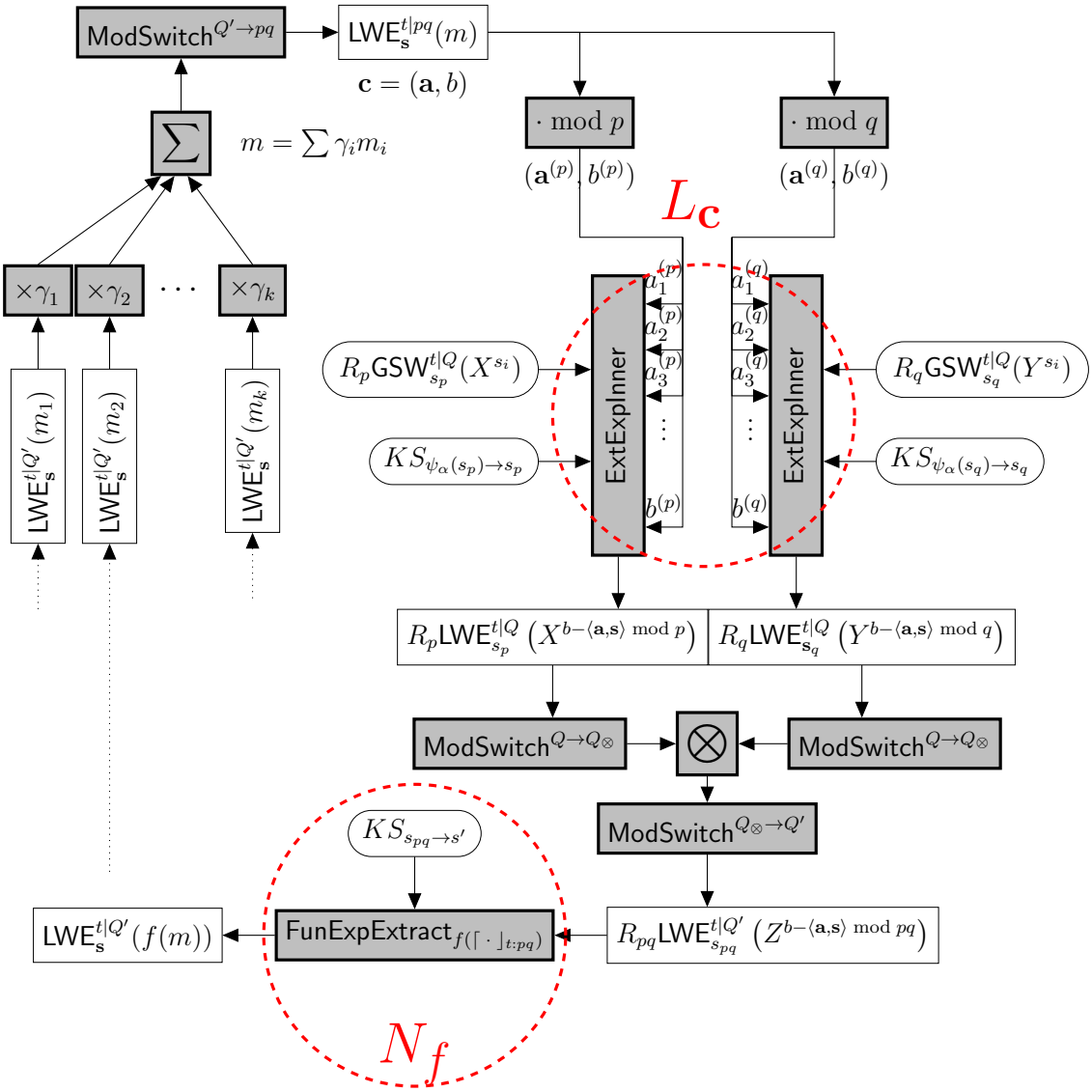
where the a_i's and b_i's' are independent uniform exponents modulo d. One notes that this distribution is invariant by permutation over $\{1, T, \ldots, T^{d-1}\}$: we have preserved the symmetries of the ring. Note that this procedure would get rather slow for large σ, yet we won't exceed $\sigma \leq 8$ in our parameter choices.

The above procedure would not be appropriate for composite degree d, as more care is required to construct a lift as done in section 6.2.2. Yet, while we will make use of circulant ring R_d with composite degree $d = pq$, we will never directly construct ciphertexts over that ring. Indeed, the ciphertext in R_{pq} will be publicly constructed by tensoring two ciphertexts from R_p and R_q, and are therefore no easier to decrypt than the original ciphertexts over R_p and R_q.

6.3 Building the gate

Before going into the detail of my contribution on the implementation of the scheme, we need to present briefly the multiple building blocks that compose the gate. Figure 6.1 presents an overview of the scheme and should be used as a guidance to understand this section. Some of the blocks (KeySwitch for example) do not appear anywhere because they belong to a superior detail level. In this section we present the building blocks and in the next, the optimisation that can be done on the structure or on some precise points.

Most of the operations presented below are meaningful both in the ring/circulant-setting or over the integers. We consider the RLWE problem over rings $R_d = \mathbb{Z}[X]/(X^d - 1)$ with d



On the left side, there are the k input bits $m_1, \cdots, m_k$ that get combined into $m \in \mathbb{Z}_t$. On the right side we have the two Homomorphic Accumulator ExtExpInner, which perform the linear part $L_{\mathbf{c}}$ of the bootstrapped computation in a CRT fashion. After tensoring it is fed to the non-linear part of the computation $N_f : x \mapsto f(\lfloor tx/q \rceil \bmod t)$, *i.e.* FunExpExtract, where f is the function to be homomorphically evaluated. The computation is intrinsically done with the bootstrapping process, so the final output can directly be used as input.
Grey boxes represent operations, white square boxes represent ciphertexts, and rounded white boxes represent key material. The linear step $L_{\mathbf{c}}$ and the non-linear step N_f discussed in the introduction are highlighted by dashed red circles.

Figure 6.1 – Scheme overview.

prime and over $R = \mathbb{Z}$ (i.e. simply the LWE problem). However, most of the results presented in this section also hold for cyclotomic rings. We assume that coefficients of ring elements in R/QR can be added and multiplied in constant time since, in our implementation, each coefficient fits into a machine word. Thus, adding two ring elements takes time $O(d)$ and multiplying them takes time $O(d \log d)$ using FFT.

Below we omit the proofs of the algorithms because they are the work of Max Fillinger and involve some theory on sub-gaussian random variables which are outside the scope of our work. They can be found in the published version [BDF17].

6.3.1 Known building blocks

Let us first recall, within our formalism, known building blocks from the literature. The only novelty in this section concerns the FunExpExtract function: while this was already constructed in previous work, in our setup we will need to apply a trick from [GHPS12] to improve its efficiency.

Linearity

$$\mathsf{Add} : R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m;\, E) \times R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m';\, E') \to R_d\mathsf{LWE}_{s}^{t|Q}\left(m + m';\, \sqrt{E^2 + E'^2}\right)$$

The Add operations are computed by simply adding the ciphertexts component-wise. Hence Add has a time complexity of $O(nd)$. The error term in the result of Add holds when the error terms in the input ciphertexts are independent. Otherwise, it is $E + E'$.

$$x \in R_d, \mathsf{Mult}_x : R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m;\, E) \to R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}\,(xm;\, |x|E)$$

The Mult_x operations work by scalar multiplication with x. The runtime is $O(nd \log d)$.

Modulus switching

In our scheme we need several times to change the ciphertext modulus.

$$\mathsf{ModSwitch}^{Q\to Q'} : R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m;\, E) \to R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q'}\left(m;\, \sqrt{(kE)^2 + 1 + \sum_i |\mathbf{s}_i|^2}\right)$$

The basic idea of modulus switching is to multiply the ciphertext with Q'/Q, or rather $\lfloor Q'/t \rceil$ / $\lfloor Q/t \rceil$. The runtime is $O(d)$.

In practice, we only use modulus switching in the following two cases: when the dimension of the key is $n = 1$, and for short keys in $\mathbb{Z}^n$, i.e. n-dimensional keys where $|\mathbf{s}_i| \leq 1$. In the first case, the error parameter simplifies to $\sqrt{(kE)^2 + 1 + |s|^2}$, in the second case to $\sqrt{(kE)^2 + n + 1}$.

Key switching

In our scheme, we also change the key under which the ciphertexts are encrypted.

$$\mathsf{KeySwitch}_{\mathbf{S}}^{\mathbf{s}\to s'} : R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m;\, E) \to R_d\mathsf{LWE}_{s'}^{t|Q}\left(m;\, \sqrt{E^2 + \sigma^2 d^2 nK}\right).$$

Algorithm 6.1 $\mathsf{KeySwitch}_{\mathbf{S}}^{\mathbf{s}\to s'}(\mathbf{c})$: Transforms an $R_d\mathsf{LWE}$ ciphertext under key $\mathbf{s}$ into a ciphertext under s'.

Require:
$\mathbf{S} = [\mathbf{S}_i]_{i\in[n]}$ where $\mathbf{S}_i \in R_d\mathsf{LWE}_{s'}^{Q|Q}(s_i \cdot \mathbf{g}^T)$.
A ciphertext $(\mathbf{a}, b) \in R_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m)$ for some $m \in R/tR$.
Ensure: A ciphertext $c \in R_d\mathsf{LWE}_{s'}^{t|Q}(m)$

return $(\mathbf{0}_{n'}, b) - \mathbf{g}^{-T}(\mathbf{a}) \cdot \mathbf{S}$

As it is the case for FV and SHIELD (see previous chapter), the choice of the basis decomposition B for the gadget is important. It allows to trade off key size and running time against error growth. We use

$$S = \left[R_d\mathsf{LWE}_{s'}^{1,Q}(B^j \mathbf{s}_i; \sigma)\right]_{i=1\ldots n, j=0\ldots K-1}, \text{with} K = \lceil \log_B Q \rceil$$

as key material. The key size decreases to $O(nn'dK \log Q)$, and the running time decreases to $O(d \log dnn'K)$, while the output error parameter also increases to $\sqrt{E^2 + \sigma^2 d^2 B^2 nK}$.

External Multiplication

Similarly as in FHEW and TFHE, we perform multiplication between $R_d\mathsf{LWE}$ and $R_d\mathsf{GSW}$ ciphertexts. The runtime of this operation is $O(Kd \log d)$.

$$\begin{aligned}\mathsf{ExtMult}: \quad & R_d\mathsf{LWE}_{s}^{t|Q}(T^m;\, E) \times R_d\mathsf{GSW}_{s}^{t|Q}(T^{m'};\, E') \\ & \to R_d\mathsf{LWE}_{s}^{t|Q}\left(T^{m+m'};\, \sqrt{E^2 + 2Kd^2E'^2}\right)\end{aligned}$$

Algorithm 6.2 $\mathsf{ExtMult}(\mathbf{c}, \mathbf{C})$: Multiplies an $R_d\mathsf{LWE}$ ciphertext and a $R_d\mathsf{GSW}$ ciphertext into a $R_d\mathsf{LWE}$ ciphertext.

Require: A ciphertext $\mathbf{c} \in R_d\mathsf{LWE}_{s}^{t|Q}(T^m)$, and a ciphertext $\mathbf{C} \in R_d\mathsf{GSW}_{s}^{t|Q}(T^{m'})$.
Ensure: A ciphertext $\mathbf{c}' \in R_d\mathsf{LWE}_{s}^{t|Q}\left(T^{m+m'}\right)$.

return $\mathbf{G}^{-1}\left(\lfloor Q/t \rceil^{-1} \cdot \mathbf{c}\right) \cdot \mathbf{C}$

Function extraction in the exponent

To construct our bootstrapped scheme, we need to somehow evaluate a function on the ciphertext and get the result back. In our scheme, this is achieved with $\mathsf{FunExpExtract}$. It takes what comes out of the homomorphic accumulator.

$$\mathsf{FunExpExtract}_{F,\mathbf{S}}^{s_{pq}\to\mathbf{s}}: \quad R_{pq}\mathsf{LWE}_{s_{pq}}^{t|Q'}(Z^m;\, E) \\ \to \mathsf{LWE}_{\mathbf{s}}^{t|Q'}\left(F(m);\, |F|\sqrt{E^2+3\sigma^2p^2q^2K}\right)$$

for some function $F:\mathbb{Z}_{pq}\to\mathbb{Z}_t$ where $|F|=\sum_{i\in\mathbb{Z}_{pq}}|F(i)|$ and $\mathbf{s}\in\mathbb{Z}^p$.

Let us first consider the function F_0 that maps $0\mapsto 1$ and $k\mapsto 0$ for $k\neq 0$. If we can extract this function, we can extract *any* function by first multiplying the ciphertext with an appropriate polynomial. This extraction is easily provided by the trace function $\mathrm{Tr}^*=\mathrm{Tr}^*_{R_{pq}/\mathbb{Z}}$. Indeed, if $(a,b)\in R_{pq}\mathsf{LWE}_s(m)$, then $(\mathbf{a},\mathrm{Tr}^*(b))\in\mathsf{LWE}_{\mathbf{s}}(m_0)$, where $\mathbf{a},\mathbf{s}\in\mathbb{Z}^{pq}$ are the vectors of coefficients of a and s.

However, this leads to an LWE ciphertext with quadratic dimension $pq=\Theta(n^2)$, that must be key-switched to a much smaller dimension $\Theta(n)$. Such a key-switch without any ring structure would require up to $\tilde{\Theta}(n^3)$ running time, and as much key material.

To circumvent this issue, we exploit the intermediate ring, following one of the tricks of [GHPS12]. Namely, we choose a key in R_p, which can also be viewed as an element of R_{pq}. Exploiting the structure of R_{pq}, switching to this key requires only $\tilde{\Theta}(pq)=\tilde{\Theta}(n^2)$ operations. Then, one can trace a down to R_p, and b down to $\mathbb{Z}$, and obtain the desired result.

Algorithm 6.3 $\mathsf{FunExpExtract}_{F,\mathbf{S}}^{s_{pq}\to\mathbf{s}}$: Turns an $R_{pq}\mathsf{LWE}$ encryption of Z^m into an LWE encryption of $F(m)$.

Require:
A ciphertext $\mathbf{c}\in R_{pq}\mathsf{LWE}_{s_{pq}}^{t|Q'}(Z^m)$,
A function $F:\mathbb{Z}_{pq}\to\mathbb{Z}_t$,
A key switching key $\mathbf{S}$ from s_{pq} to $s'\in R_p\subseteq R_{pq}$, where $s'=\sum_{i=0}^{p-1}(s_{pq})_{i+1}X^i$.
Ensure: A ciphertext $c'\in\mathsf{LWE}_{\mathbf{s}}^{t|Q'}(F(m))$.

$f\leftarrow\sum_{i\in\mathbb{Z}_{pq}}F(i)Z^{-i \bmod pq}\in R_{pq}$
$\mathbf{c}\leftarrow\mathsf{KeySwitch}_{\mathbf{S}}^{s_{pq}\to s'}(\mathbf{c})$ ▷ $\in R_{pq}\mathsf{LWE}_{s'}^{t|Q'}(Z^m)$
$\mathbf{c}\leftarrow\mathsf{Mult}_f(\mathbf{c})$ ▷ $\in R_{pq}\mathsf{LWE}_{s'}^{t|Q'}(\sum_{i\in\mathbb{Z}_{pq}}F(i)Z^{m-i \bmod pq})$
$(a,b)\leftarrow\mathrm{Tr}^*_{R_{pq}/R_p}(\mathbf{c})$ (component-wise) ▷ $\in R_p\mathsf{LWE}_{s'}^{t|Q'}(\sum_{i,\text{ st }q|(m-i)}F(i)X^{m-i \bmod pq})$
$\mathbf{a}\leftarrow(a_0,a_{p-1},a_{p-2},\ldots,a_1)$
$b\leftarrow\mathrm{Tr}^*_{R_p/\mathbb{Z}}(b)$
return $(\mathbf{a},b)$

Algorithm 6.3 runs in time $O(pq\log(pq)\log Q')$. A careful look at the proof shows that we could reduce the error growth by performing the multiplication before the key-switch. However, doing the key-switch first allows to amortise the cost of gates with multiple outputs.

Indeed the algorithm consists in computing

$$c_{pq} \mapsto \mathrm{Tr}^*_{R_{pq}/R_p}(f \cdot \mathbf{G}^{-T}(c_{pq}) \cdot \mathbf{S})$$

where the most expensive part of the computation $\mathbf{G}^{-T}(c_{pq}) \cdot \mathbf{S}$ (i.e. the KeySwitch) can be re-used for several different f's. Hence, at the cost of one KeySwitch we can extract multiple functions on the same input bits. This allows to construct more evolved gates with negligible extra costs.

6.3.2 New building blocks

In this part, we present the new elementary blocks that were introduced for this scheme.

Exponent Multiplication by Galois Conjugation

For $\alpha \in \mathbb{Z}_d^*$, we let ψ_α be the automorphism of R_d defined by $T \mapsto T^\alpha$. We define a Galois Conjugation on ciphertexts

$$\mathsf{Galois}: \quad R_d\mathsf{LWE}_s^{t|Q}(T^m;\, E) \to R_d\mathsf{LWE}_{\psi_\alpha(s)}^{t|Q}(T^{\alpha m};\, E)\,.$$

Given a R_dLWE-ciphertext $(a, as + \lfloor Q/t \rceil\, T^m + e)$, by applying ψ_α component-wise, we obtain $(\psi_\alpha(a), \psi_\alpha(a) \cdot \psi_\alpha(s) + \lfloor Q/t \rceil\, T^{\alpha m} + \psi_\alpha(e))$. The running time is $O(d)$, because for $x \in R$ $\psi_\alpha(x)$ is computed simply by permuting the coefficients of x. Even if the ciphertext is in FFT representation, the runtime remains $O(d)$, as ψ_α also acts on those representations by permutation.

CRT in the exponent by tensoring

A key trick from our scheme is that of combining the computation of two small (and fast) homomorphic accumulators into one result. This is the purpose of the ExpCRT operation.

$$\begin{aligned}\mathsf{ExpCRT}: \quad & R_p\mathsf{LWE}_{s_p}^{t|Q_\otimes}(X^{m_p};\, E_p) \times R_q\mathsf{LWE}_{s_q}^{t|Q_\otimes}(Y^{m_q};\, E_q) \\ & \to R_{pq}\mathsf{LWE}_{s_{pq}}^{t|Q_\otimes}\left(Z^m;\, \sqrt{E_p^2 + E_q^2} + t\sqrt{2\lambda} E_p E_q\right)\end{aligned}$$

where $m = \alpha m_p + \beta m_q$ is such that $m_p = m \bmod p$ and $m_q = m \bmod q$ and $s_{pq} = (-\psi_\alpha(s_p) \otimes \psi_\beta(s_q), \psi_\alpha(s_p) \otimes 1, 1 \otimes \psi_\beta(s_q))$.

Note that we need $t \cdot \lfloor Q_\otimes / t \rceil = 1$ for the algorithm to be correct. This condition can be easily satisfied in our bootstrapping scheme because we perform a modulus switch before and after ExpCRT. Algorithm 6.4 runs in time $\Theta(pq)$.

Algorithm 6.4 $\mathsf{ExpCRT}(\mathbf{c}_p, \mathbf{c}_q)$

Require: Ciphertexts $\mathbf{c}_p \in R_p\mathsf{LWE}^{t|Q_\otimes}_{s_p}(X^{m_p}; E_p)$ and $\mathbf{c}_q \in R_q\mathsf{LWE}^{t|Q_\otimes}_{s_q}(Y^{m_q}; E_q)$.
Ensure: A ciphertext $\mathbf{c} \in R_{pq}\mathsf{LWE}^{t|Q_\otimes}_{s_{pq}}(Z^m)$

$(a_p, b_p) \leftarrow \mathsf{Galois}^\alpha(\mathbf{c}_p)$ ▷ $\in R_p\mathsf{LWE}^{t|Q_\otimes}_{\psi_\alpha(s_p)}(X^{\alpha m_p}; E_p)$
$(a_q, b_q) \leftarrow \mathsf{Galois}^\beta(\mathbf{c}_q)$ ▷ $\in R_q\mathsf{LWE}^{t|Q_\otimes}_{\psi_\beta(s_q)}(Y^{\beta m_q}; E_q)$
$\mathbf{a} \leftarrow (a_p \otimes a_q, a_p \otimes b_q, b_p \otimes a_q)$
return $(t\mathbf{a}, tb_p \otimes b_q)$

External Multiply-and-Add in the exponent

We construct an (External) Multiply-and-Add operation in the exponent, for a public coefficient $\alpha \in \mathbb{Z}_d^*$.

A similar speed-up was obtained in [CGGI16] using a different technique, namely a *Mux* operation. We are unfortunately unable to use it in our circulant setup, essentially because encryptions of 0 are not allowed: our IND-CPA-security guarantee only applies to encryptions of X^m for some $m \in \mathbb{Z}_d$. Yet, our technique is more general, precisely we do not restrict the secret input vector to have binary coefficients.

$$\mathsf{ExtExpMultAdd}: \quad R_d\mathsf{LWE}^{t|Q}_s(T^{m'};\, E) \times R_d\mathsf{GSW}^{t|Q}_s(T^m;\, E') \to R_d\mathsf{LWE}^{t|Q}_s\left(T^{\alpha m+m'};\, E''\right).$$

Algorithm 6.5 $\mathsf{ExtExpMultAdd}^\alpha_{\mathbf{S}^\alpha,\mathbf{S}^\beta}(\mathbf{c}, \mathbf{C})$

Require: $\alpha \in \mathbb{Z}_d^*$, with inverse $\beta = \alpha^{-1} \in \mathbb{Z}_d^*$
A $\psi_\alpha(s) \to s$ key switching key $\mathbf{S}^\alpha \in R_d\mathsf{LWE}^{Q|Q}_s\left(\psi_\alpha(s)\cdot\mathbf{g}^T;\, \sigma\right)$
A $\psi_\beta(s) \to s$ key switching key $\mathbf{S}^\beta \in R_d\mathsf{LWE}^{Q|Q}_s\left(\psi_\beta(s)\cdot\mathbf{g}^T;\, \sigma\right)$
A ciphertext $\mathbf{c} \in R_d\mathsf{LWE}^{t|Q}_s(T^{m'};\, E)$. A ciphertext $\mathbf{C} \in R_d\mathsf{GSW}^{t|Q}_s(T^m;\, E')$
Ensure: A ciphertext $\mathbf{c}' \in R_d\mathsf{LWE}^{t|Q}_s(T^{\alpha m+m'};\, \sqrt{E^2 + d^2K(4\sigma^2 + E'^2)})$.

$\mathbf{c}_1 \leftarrow \mathsf{Galois}^\beta(\mathbf{c})$ ▷ $\in R_d\mathsf{LWE}^{t|Q}_{\psi_\beta(s)}\left(T^{\beta m'}\right)$
$\mathbf{c}_2 \leftarrow \mathsf{KeySwitch}^{\psi_\beta(s)\to s}_{\mathbf{S}^\beta}(\mathbf{c}_1)$ ▷ $\in R_d\mathsf{LWE}^{t|Q}_s\left(T^{\beta m'}\right)$
$\mathbf{c}_3 \leftarrow \mathsf{ExtMult}(\mathbf{C}, \mathbf{c}_2)$ ▷ $\in R_d\mathsf{LWE}^{t|Q}_s\left(T^{m+\beta m'}\right)$
$\mathbf{c}_4 \leftarrow \mathsf{Galois}^\alpha(\mathbf{c}_3)$ ▷ $\in R_d\mathsf{LWE}^{t|Q}_{\psi_\alpha(s)}\left(T^{\alpha m+m'}\right)$
$\mathbf{c}_5 \leftarrow \mathsf{KeySwitch}^{\psi_\alpha(s)\to s}_{\mathbf{S}^\alpha}(\mathbf{c}_4)$ ▷ $\in R_d\mathsf{LWE}^{t|Q}_s\left(T^{\alpha m+m'}\right)$
return $\mathbf{c}_5$.

External Inner-product in the Exponent

By chaining, this allows us to evaluate inner products $\langle \mathbf{x}, \mathbf{y}\rangle$ over $\mathbb{Z}_d$ in the exponent, given GSW encryptions $R_d\mathsf{GSW}^{t|Q}_s(T^{x_i})$ and a public vector of coefficients $\mathbf{y} \in \mathbb{Z}_d^\ell$. This procedure

allows evaluation of inner products in exponents with $\log d$ times less homomorphic additions in exponents than in FHEW, also less key material. Algorithm 6.6 runs in time $\Theta(lKd\log d)$.

$$\mathsf{ExtExpInner}^{\mathbf{y}}_{[\mathbf{S}^\alpha]_\alpha}: \quad \bigoplus_{i=1}^{\ell} R_d\mathsf{GSW}^{t|Q}_s(T^{x_i};\, E') \to R_d\mathsf{LWE}^{t|Q}_s\left(T^{\langle \mathbf{x},\mathbf{y}\rangle};\, \sqrt{2K\ell^2d^2\sigma^2 + 2K\ell d^2E'^2}\right).$$

Algorithm 6.6 $\mathsf{ExtExpInner}^{\mathbf{y}}_{[\mathbf{S}^\alpha]_\alpha}([\mathbf{C}_i]_{i\in[l]})$

Require: A public vector $\mathbf{y} \in \mathbb{Z}_d^\ell$
A $\psi_\alpha(s) \to s$ key switching key $\mathbf{S}^\alpha \in R_d\mathsf{LWE}^{Q|Q}_s\left(\psi_\alpha(s)\cdot\mathbf{g}^T\right)$ for each $\alpha \in \mathbb{Z}_d^*$
A ciphertext $\mathbf{C}_i \in R_d\mathsf{GSW}^{t|Q}_s(T^{x_i})$ for each $i \in [\ell]$
Ensure: A ciphertext $\mathbf{c} \in R_d\mathsf{LWE}^{t|Q}_s\left(T^{\langle \mathbf{x},\mathbf{y}\rangle}\right)$.

$\mathbf{c} \leftarrow (0, T^0)$ ▷ $\in R_d\mathsf{LWE}^{t|Q}_s(T^0; 0)$
for i from 1 to ℓ where $\mathbf{y}_i \neq 0$ **do**
  $\alpha = \mathbf{y}_i$; $\beta = \alpha^{-1} \bmod d$
  $\mathbf{c} \leftarrow \mathsf{ExtExpMultAdd}^{\alpha}_{\mathbf{S}^\alpha,\mathbf{S}^\beta}(\mathbf{c}, \mathbf{C}_i)$ ▷ $\in R_d\mathsf{LWE}^{t|Q}_s\left(T^{\sum_{j=1}^{i}\mathbf{x}_j\mathbf{y}_j}\right)$
end for
return c

In order to make things faster we also introduced (and implemented) two tricks.

Factoring Galois-KeySwitch sequences. We note that it is possible to factor some operations when chaining $\mathsf{ExtExpMultAdd}^\alpha$ and $\mathsf{ExtExpMultAdd}^\beta$, by applying $\mathsf{Galois}^{\alpha\beta^{-1}}$ rather than Galois^α followed by Galois^β (together with the appropriate key-switches), as illustrated in Fig 6.2. The implementation follows this approach.

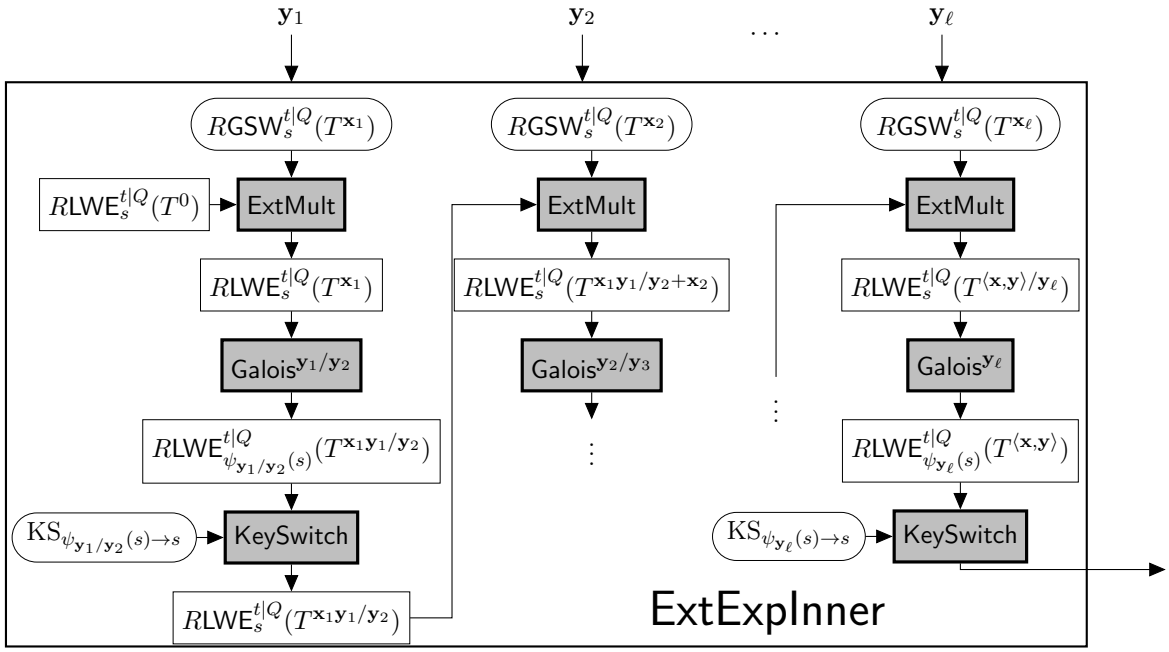
Furthermore, if $\mathbf{y} \in \mathbb{Z}_d^\ell$ contains repeated values, it is possible to re-index the inner product to make equal values contiguous, and skip useless Galois^1 operations. Those tricks also decrease the final error E by constant factors. In practice we noticed that this re-indexing was more expansive than the gain. It may be possible to achieve a positive speed-up with this trick but we would need to look more closely at its implementation.

Pushing this trick to its limits, if ℓ is large enough, one could re-index the inner product so that the $\alpha\beta^{-1}$ all belong to a small subset $\mathbb{Z}_d^*$ (of size roughly $d/\ell+2$, assuming the public vector $\mathbf{y} \in \mathbb{Z}_d^\ell$ is uniformly random), allowing to decrease the size of the key material. In combination with the following optimisation, this should lead to reduce the overall key size by a significant factor. We did not attempt to implement this yet.

Decreasing LWE dimension. In our theoretical scheme, the homomorphic inner product in exponent operation is performed over vectors of length $\ell = p+1$ where p is the dimension of the secret in the LWE scheme.

In practice, we remark that this dimension is quite larger than needed for security, given the amount of noise and the (small) modulus pq of those ciphertexts. We therefore

Figure 6.2 – Optimised ExtExpInner (External Inner Product in Exponent) overview



proceed with an extra LWE key-switch just after the combination of the LWE ciphertexts. In practice it allows to decrease the dimension by a factor between 2 and 3, which accelerates the ExtExpInner operations by the same factor. As a small added bonus, it also slightly decreases the error in the ciphertexts output by this function.

6.3.3 Joining the building blocks

In this section, we explain how the building blocks described above fit together to form the homomorphic evaluation and bootstrapping procedure EvalBootstrap. See Fig 6.1 for a schematic overview. We build an algorithm that, given ciphertexts $\mathbf{c}_i \in \mathsf{LWE}_{\mathbf{s}}(m_i; E_{\text{in}})$, $i \in \{1, \ldots, k\}$ with $\mathbf{s} \in \mathbb{Z}_{Q'}^p$ a short vector (i.e. $\mathbf{s}_i \in \{-1, 0, 1\}$ for all i), a function $f : \mathbb{Z}_t \to \mathbb{Z}_t$, and coefficients $\gamma_1, \ldots, \gamma_k \in \mathbb{Z}_t$ such that $\sum_i |\gamma_i| \leq t$, produces $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}(f(m); E_{\text{out}})$ where $m = \sum_{i=1}^k \gamma_i m_i$. We use the following parameters for the building blocks:

- n as the security parameter,
- $p, q = \Theta(n)$, $Q = \text{poly}(n)$, $K = \lceil \log Q \rceil = O(\log n)$, $t = \Theta(n)$ such that $t \leq \sqrt{pq}/4$,
- $\lambda = \Theta(n)$, such that $\lambda \leq q$ as the failure parameter; the decryption and homomorphic evaluation procedures should only fail with probability exponentially small in λ,
- σ as the error parameter used in the key material,
- $Q', Q_\otimes = O(Q/\sqrt{n\sigma})$, with $t \cdot \lfloor Q_\otimes/t \rceil = 1 \bmod Q_\otimes$.

For $m_i \in \{0,1\}$, the algorithm can evaluate arbitrary k-bit gates if $t \geq 2^k$, using $\gamma_i = 2^{i-1}$ and an appropriately chosen f. We can compute a threshold gate if $t > k$ by setting $\gamma_i = 1$ for all i.

Algorithm 6.7 $\mathsf{EvalBootstrap}_{\mathcal{S}}^{f,\gamma_1,\ldots,\gamma_k}(\mathbf{c}_1,\ldots,\mathbf{c}_k)$: Homomorphically evaluates a function and produce a bootstrapped encryption of the result.

Require: $\mathbf{c}_i \in \mathsf{LWE}_{\mathbf{s}}^{t|Q'}(m; E_{\text{in}})$, $f : \mathbb{Z}_t \to \mathbb{Z}_t$, $\gamma_i \in \mathbb{Z}_t$ where $\gamma E_{\text{in}} \leq T$ for a certain $T = \Theta(Q/(n^2\sqrt{\sigma}))$, and $\mathcal{S}$ is the required public key material consisting of:

- Bootstrapping keys $\mathbf{BK}_i^{(d)} \in R_d\mathsf{LWE}_{s^{(d)}}^{t|Q}(\mathbf{s}_i \bmod d; \sigma)$ for $i = 1,\ldots,n$ and $d = p, q$
- Key switching keys $\mathbf{S}^{d,\alpha}$ from $\psi_\alpha(s)$ to s for $d \in \{p,q\}$ and $\alpha \in \mathbb{Z}_d^*$
- A key switching key $\mathbf{S}$ from s_{pq} to s' where $s_{pq} = \big(-\psi_\alpha(s_p) \otimes \psi_\beta(s_q), \psi_\alpha(s_p) \otimes 1, 1 \otimes \psi_\beta(s_q)\big)$ for $\alpha = q^{-1} \bmod p$, and $\beta = p^{-1} \bmod q$, and $s' = \sum_{i=0}^{p-1}(s_{pq})_{i+1}X^i$

Ensure: $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}^{t|Q'}\Big(f\big(\sum_{i=1}^k \gamma_i m_i\big); E_{\text{out}}\Big)$ where $E_{\text{out}} = O\big(|f|n^{4.5}\sigma\big)$

$\mathbf{c} \leftarrow \sum_{i=1}^k \mathbf{c}_i$ $\quad \triangleright \in \mathsf{LWE}_{\mathbf{s}}^{t|Q'}(m; \gamma E)$

$\mathbf{c} \leftarrow \mathsf{ModSwitch}^{Q' \to pq}(\mathbf{c})$ $\quad \triangleright \in \mathsf{LWE}_{\mathbf{s}}^{t|pq}(m; \sqrt{r^2\gamma^2E^2 + (p+1)^2})$ where $r = \lfloor pq/t \rceil \,/\, \lfloor Q'/t \rceil$

$(\mathbf{a}_p, b_p) \leftarrow \mathbf{c} \bmod p$

$(\mathbf{a}_q, b_q) \leftarrow \mathbf{c} \bmod q$

$\mathbf{c}_p \leftarrow X^{b_p} \cdot \mathsf{ExtExpInner}_{[\mathbf{S}^{p,\alpha}]_\alpha}^{-\mathbf{a}_p}\Big(\big[\mathbf{BK}_i^{(p)}\big]_i\Big)$

$\quad \triangleright \in R_p\mathsf{LWE}_s^{t|Q}\big(X^{b-\langle \mathbf{a},\mathbf{s}\rangle \bmod p}; O\big(n^{2.5}\sigma\big)\big)$

$\mathbf{c}_q \leftarrow Y^{b_q} \cdot \mathsf{ExtExpInner}_{[\mathbf{S}^{q,\alpha}]_\alpha}^{-\mathbf{a}_q}\Big(\big[\mathbf{BK}_i^{(q)}\big]_i\Big)$

$\quad \triangleright \in R_q\mathsf{LWE}_s^{t|Q}\big(Y^{b-\langle \mathbf{a},\mathbf{s}\rangle \bmod q}; O\big(n^{2.5}\sigma\big)\big)$

$\mathbf{c}_p \leftarrow \mathsf{ModSwitch}^{Q \to Q_\otimes}(\mathbf{c}_p)$ $\quad \triangleright \in R_p\mathsf{LWE}_{s^{(p)}}^{t|Q_\otimes}\Big(X^{b-\langle \mathbf{a},\mathbf{s}\rangle \bmod p}; O\big(n\sqrt{\sigma}\big)\Big)$

$\mathbf{c}_q \leftarrow \mathsf{ModSwitch}^{Q \to Q_\otimes}(\mathbf{c}_q)$ $\quad \triangleright \in R_q\mathsf{LWE}_{s^{(q)}}^{t|Q_\otimes}\Big(Y^{b-\langle \mathbf{a},\mathbf{s}\rangle \bmod q}; O\big(n\sqrt{\sigma}\big)\Big)$

$\mathbf{c}_{pq} \leftarrow \mathsf{ExpCRT}(\mathbf{c}_p, \mathbf{c}_q)$ $\quad \triangleright \in R_{pq}\mathsf{LWE}_{s_{pq}}^{t|Q_\otimes}\big(Z^{b-\langle \mathbf{a},\mathbf{s}\rangle}; O\big(n^{3.5}\sigma\big)\big)$

$\mathbf{c}_{pq} \leftarrow \mathsf{ModSwitch}^{Q_\otimes \to Q'}\big(\mathbf{c}_{pq}\big)$ $\quad \triangleright \in R_{pq}\mathsf{LWE}_{s_{pq}}^{t|Q'}\big(Z^{b-\langle \mathbf{a},\mathbf{s}\rangle}; O\big(n^{3.5}\sigma\big)\big)$

$F \leftarrow (x \mapsto f(\lfloor tx/q \rceil \bmod t))$ $\quad \triangleright F : \mathbb{Z}_{pq} \to \mathbb{Z}_t$, $|F| = |f|pq/t = O(|f|n)$

$\mathbf{c} \leftarrow \mathsf{FunExpExtract}_{F,\mathbf{S}}^{s_{pq} \to \mathbf{s}}\big(\mathbf{c}_{pq}\big)$ $\quad \triangleright \in \mathsf{LWE}_{\mathbf{s}}^{Q'|t}\big(f(m); O\big(|f|n^{4.5}\sigma\big)\big)$

return $\mathbf{c}$

6.3.4 Heuristic error propagation

Our theoretical analysis of the scheme used subgaussian analysis to provide bounds on error propagation that are already significantly better than worst-case bounds. Yet those bounds are asymptotic, without explicit constants, and for some operations may not be perfectly tight. As in previous work [DM15, CGGI16], when it comes to choosing practical parameters, we rely on a tighter but heuristic analysis of error propagation, essentially treating all random

variables as independent gaussians. More precisely, considering that the critical random variable for correctness is obtained as the sum of many random variables, we only compute its variance as the sum of the variances of its terms, and treat this final result as gaussian in accordance with the central limit theorem (which is formally not applicable due to potential dependencies).

We refer to the original publication [**?**] for these better bounds that were derived by Léo and Max. We could successfully confirm all these heuristic equations by measuring the actual errors in our implementation.

6.4 Implementation

Now that we have presented the scheme in details, we can turn to the implementation challenges that constitute my contribution to this work. When I arrived at CWI in March 2017, the main aspects of the scheme were already there, based on Léo's and Max's works. We already had the overall Figure 6.1 in place. It remained to clear out some aspects such as proofs, practical error propagation, choice of parameters and implementation performance and tweaks. Working on this implementation kept me busy for my three months there. We shall present now the details of this contribution.

In the end, we developed a complete implementation of the scheme in C++11. Our objective was to make it efficient and easily usable by fellow cryptographers and users. The performance we got enabled us to homomorphically evaluate binary gates on six input bits in roughly 6.4 seconds on a regular laptop.

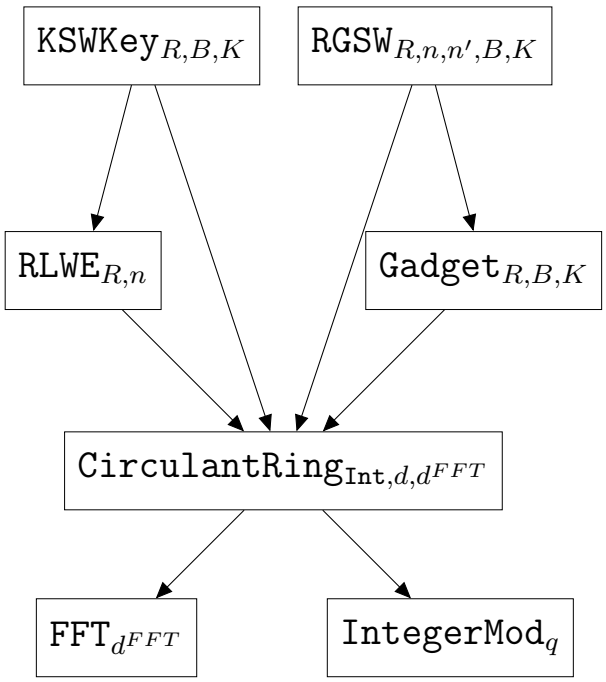
6.4.1 Data representation

As we saw above, there are several *objects* (in the board sense) to handle: different ciphertexts (LWE, $R_d\mathsf{LWE}$, $R_d\mathsf{GSW}$, ...), key materials (for encryption, key switching) and so on. To handle all this, we need to be able to have circulant ring elements in R_d whose dimension d takes different values: p, q in the accumulator, pq in the function extraction or even 1 for the input/output of the gate. Also, all arithmetic operations are performed modulo different integers $Q, Q', Q_\otimes \ldots$ We put in Figure 6.3 the full class diagram where each class has in subscript the parameter upon which it depends.

Low-level classes

At the bottom, we developed two classes: `IntegerMod` and `CirculantRing` to handle modular arithmetic and ring operations respectively. For performance reasons, we create the different integers and ring elements thanks to the template mechanism. As a reminder, the advantage of doing so is that the compiler will generate different binary code for all the different instances, so each type is optimised by the compiler. For example, modular arithmetic done with `%` operator modulo some power of 2 gets rewritten automatically with shifts by the compiler. The downside is a longer compilation time.

The `IntegerMod` class is then defined with one template attribute and contains one `int64_t` field to hold the value. `CirculantRing` has two template parameters: the class



R stands for a ring; d is the size of a ring; d^{FFT} the FFT dimension for ring element multiplication; n and n' are numbers of ring elements; B and K are a decomposition basis and size.

Figure 6.3 – Class diagram. The relation $A \to B$ reads "A needs B"

for its coefficients (the ring does not know, and does not need to, the modulus) and the dimension of the ring. It has one dynamically allocated array of elements.

Fast multiplication with FFT

To perform the numerous FFTs of the scheme, we rely on a third party library FFTW [FJ05]. It is a well-established, free and open-source, library with excellent performances so we provide a connector class `FFT` to handle the library calls. Working with FFTW implies that at the beginning of the programme some *plans* are computed (to be used to perform the FFT transforms later) with optimal performances on the computer used for execution. This *wisdom* is saved by our programme and has to be generated only once per computer. Each `CirculantRing` type (i.e. each template parameter set) has a static pointer to an FFT object to delegate the multiplication to it. We also add few static caches, that are allocated when the program starts and can be used for intermediary operations within `CirculantRing`. It saves the many allocations/deallocations that would happen otherwise and badly impact the performances.

Since we are dealing with circulant ring elements, we may wish to run the FFT operation in the ring dimension exactly. We would benefit from the Wrapped Convolution property we discussed in the previous chapter. But we need our ring dimensions to be prime, which is the worst case for FFT efficiency. We ran some benchmarks and it turned out that it was much faster to use a bigger dimension (with small prime factors), and perform the polynomial reduction afterwards. Also, we do not meet the conditions to apply NTT (our moduli are not prime), so our choice was to stick with regular FFT computations. As such, each `CirculantRing` class has another template parameter: the FFT dimension.

More challenges arose with FFT computations since our biggest modulus is $Q = 2^{56}$ and the FFT works with *double precision* numbers (i.e. 53 bits mantissa). So we have to split the ring coefficients into two halves of 28 bits each (least and most significant parts) and apply the FFT transformation on each to prevent rounding errors and then perform the term by term multiplications correctly (à la Karatsuba). We perform this splitting trick only when needed, i.e. when the ring element at stake is not small. For example, in ExtMult products of ring elements are computed where one of them is the output of a Gadget decomposition. This operand does not need to be split before FFT forward transform because it is very small.

Gadget decomposition

For this aspect, we also created a dedicated class `Gadget`. The decomposition depends on the type to decompose, on the basis B used for the decomposition and on the size K of such decomposition. Again, we put these three parameters as template on the class. It consists only of two functions: one to generate the Gadget matrix used in R_dGSW and the other to perform the actual $\mathbf{g}^{-T}$ decomposition.

Predicates

The main objective of our scheme remains to evaluate a function on the input bits. So we provide the generic class to describe such a function $f : \mathbb{Z}_t \to \mathbb{Z}_t$. Any user function can be described by derivation of the base class and the description can take any form (look-up table (LUT), linear expression, ...).

Encryptions

Using again the power of templates, we construct one generic `RLWE` class, taking two template parameters: the ring type of its elements and the *size* of the ciphertext. This size is that of the key which can be 1 in the case of a usual Ring-LWE ($\mathbf{s}$ is one ring element of size d) or many for LWE encryptions ($\mathbf{s}$ is n integers). Also, we have the case of *Matrix-Ring-LWE* as it happens in our scheme after the ExpCRT operation: the ciphertexts are then composed of 3+1 ring elements and the associated key s_{pq} has also three components.

All this complexity is hidden to the user in the template parameters and the class `RLWE` has only two fields: an array $\mathbf{a}$ of ring elements (dynamically allocated with n elements) and another ring element b.

For R_dGSW, the ciphertexts are a matrix of size $nK \times n$ of ring elements. So several template parameters are put in place for the `RGSW` class: the ring type, the size n, the decomposition basis B and size K. A static field links each type to the adequate gadget matrix computed by the `Gadget` interface.

Key switching keys

The last non-trivial component is the `KSWKey` class. Remember from the description of the building blocks above that a key switching key consists in several R_dLWE encryptions of multiples of a key under another key. Hence, our `KSWKey` class takes several template parameters: the ring type like the others, the dimension of the source key n, that of the destination key n' and the decomposition basis B and size K. The object itself contains one field: a $n \times K$ array of R_dLWE encryptions (of size n').

Further optimisations

Pre-computations. In order to minimise the evaluation time of the gate, a maximum of heavy computations is done in the setup phase. Consequently all key materials: bootstrapping keys, key switching keys, among others, are computed ahead of time and in FFT domain. Our `CirculantRing` class allows to transparently manipulate ring element in FFT or coefficient representation which greatly contributes to both performance and code readability.

Compiler options. We went into great lengths at optimising the algorithms and the different function calls and so on. We dived into the details with `callgrind` for precise function profiling to target our efforts. We also tweaked the optimisation flags of `g++` to save up 5% more time than the usual `-O2`, which already saves 75% when compared to no flag at all.

The selected flags are: `-Ofast`, `-march=native`, `-mtune=native`, `-fno-schedule-insns`, `-funroll-loops`, `-ffinite-math-only`.

Open-source

Many efforts have also been made for general availability and usability. The whole code is documented with Doxygen and many unitary tests are provided. With under 4,000 lines of code, it remains accessible to whoever wants to tweak or improve it. The implementation is publicly available and open-source[2].

Tests. We provide extensive tests for every operations happening in the gate. They can be used for correctness checks, when playing with parameters for instance. We also provide statistical tests that allow to measure the output error for every steps in the scheme. They allowed us to confirm the heuristic error growth equations of Section 6.3.4.

6.4.2 Tweaking the parameters

Error growth and correctness. To choose the parameters, we simulated the error growth throughout the gate, using heuristic error propagation assumptions, described in Section 6.3.4. We compared the predicted variance of each step to the experimental one, and found them to be quite close. From the final variance, and according to a central limit heuristic, we predict a failure probability of only 2^{-74} for the parameter set below. In practice we have tested our scheme hundreds of time on different inputs, and never observed failure.

Security. To estimate the concrete security of our parameter set, we use the `lwe-estimator` from Albrecht [APS15]. All the LWE instances behind our LWE, $R_p\mathsf{LWE}$, $R_q\mathsf{LWE}$ ciphertexts given as part of the evaluation key offers at least 100 bits of security. This is according to the estimator as of commit `cc5f6e8`, which includes the latest result of [Alb17] for small secrets. Therefore we feel safe to claim at least 80 bits of security.

Parameter sets. For our first implementation, we targeted a 6-bit gate. The parameters of the scheme are as follows:

- For 6 input bits, the plaintext modulus $t = 2^6$.
- The ring dimensions p and q are 1439 and 1447, so $pq = 2,077,892$.
- Hence the FFT dimensions are $d_1^{FFT} = 3072 = 3 \cdot 2^{10}$ for R_p,R_q and $d_2^{FFT} = 4,194,304 = 2^{22}$ for R_{pq}. Using 3072 gives us better results than 4096 despite it not being a power of two. This did not happen for d_2^{FFT}.
- The modulus in $\mathsf{ExtExpInner}$ and the LWE are $Q, Q' = 2^{56}$.

[2] `https://github.com/gbonnoron/Borogrove`

- Errors and secrets are sampled according to Section 6.2.4. Secrets are ternary, one third of the coefficients are set to -1, another to 1 and the rest to 0. Errors have variance 4.
- For ExpCRT we want a small inverse to $\lfloor Q_\otimes/t \rceil \mod Q_\otimes$. Hence we choose $Q_\otimes = (2^{19} - t + 1)^2$ which gives us

$$\lfloor Q_\otimes/t \rceil^{-1} = -t \mod Q_\otimes$$

- Finally, for the gadget decomposition we use $B = 2^8$ and $K = 7$ for ExtExpInner and FunExpExtract and their key material.

We also have extra parameters related to the optimisation ExtExpInner. Namely, we apply an extra KeySwitch over the LWE ciphertext to decrease its length l of the inner-product from $l = p = 1439$ down to $l = 600$. This key-switch happens with modulus $Q = 2^{56}$, error standard deviation 2^{33}, and gadget parameters $B = 2^6$, $K = 10$.

6.4.3 Performances

Time

We run our test on a punchy laptop: Core i7-6500U (2.50 GHz, 4MB L2 cache), 16 GB RAM with a GNU/Linux Fedora 26 installed on a SSD. The computation is single-threaded and we got the following timings:

- FFTW *wisdom* computation (only once per computer): 68 minutes
- Key pre-processing (once per user key pair): 38 seconds
- 6-bit input, 1-bit output gate evaluation: 6.4 seconds

The gate time breaks down into:

- 0.60 s per ExtExpInner (the two could be run in parallel);
- 4.0 s for the KeySwitch in FunExpExtract;
- and only 0.55 s for the output bit related operations.

Consequently, computing another function (1 more output bit) on the same 6 input bits would add only 0.55 s, and so on. For 6-to-6 bit gate it yields just above 10 seconds.

Memory

On the memory front, we need 9.2 GB of RAM to store all key materials and for the computation. The usage breaks down as follows

- 52 MB of secret keys $(\mathbf{s}, s_p, s_q, s_{pq})$

- 2×790 MB of bootstrapping keys in ExtExpInner (all the $R_p\mathsf{GSW}_{s_p}^{t|Q}(X^{s_i})$ and $R_q\mathsf{GSW}_{s_q}^{t|Q}(Y^{s_i})$)
- 2×950 MB of key-switching keys in ExtExpInner (all the $KS_{\psi_\alpha(s_p)\to s_p}$ and $KS_{\psi_\alpha(s_q)\to s_q}$)
- 2.9 GB for the key-switching key in FunExpExtract ($KS_{s_{pq}\to s'}$)
- 1.4 GB for the extra key-switching key on the LWE ciphertext prior to the ExtExpInner.

The input and output ciphertext size is 11.3 KB. The accumulators are 22.5 KB large each and after tensoring it is 63.5 MB.

Conclusion

In this last chapter, we detailed a new scheme, improving upon FHEW. We reported both the formal description and the implementation aspects we took care of. This contribution has been accepted to Africacrypt 2018 [**?**]. Another such improvement is the scheme TFHE [CGGI16, CGGI17]. In comparison to our work, TFHE offers better performances for generic gates. Though, our scheme is simpler: we have homogeneous inputs and outputs, so no need for bootstrapping. Our unique formalism for the function f is also simpler than the LUT and automata in TFHE.

With this work, we endorsed the last role of our journey, that of the implementer. Despite the hard competition there is around homomorphic encryption, our contribution has brought new concepts and techniques. As the one doing the actual implementation, our work spanned between the theoretical aims and the practical constraints. As with lattice reduction, this point is where we need works to close, or at least narrow, the gap between theory and practice.

The scheme presented here is in its infancy and could be extended and improved by further work, for example using RNS technique [BEHZ16]. The fourth generation may however be the last, since now the noise is constant between gates. Still, there may be algorithmic improvements since the ideas of TFHE [CGGI17] are very different from those in our scheme.

Chapter 7

Closing thoughts

Isn't it the goal of the traveller, never to be there?

We shall now wrap up the discussion about the last three years of work. Not that we intend to close the study of homomorphic encryption, but we hope we pushed forward both cryptography and cryptanalysis in this topic.

Retrospective

Started in April 2015, this journey led us first to endorse an attacker role. As a newcomer to the field, it has been a very stimulating task to try to break things down. Despite the limited reach of our new attack described in Chapter 3, the interaction with the community turned out to be very fruitful. Chapter 4 is a good illustration, about our contribution to algorithmic lattice reduction. A work that was mostly driven by the `fplll` days.

Next, within the context of the Lab-STICC, we were able to join forces, together with Vincent Migliore, to explore the practicality and the precise settings of homomorphic schemes, see Chapter 5 for all the details.

Finally, thanks to other encounters, we had the chance to join Léo Ducas and Max Fillinger at CWI for a few months. This allowed us to complete the circle and become a scheme designer and implementer, for real. Chapter 6 presents this line of work.

Our work in an evolving context

Following the trend around homomorphic encryption proved to be a real challenge. It required us to stay constantly tuned to the IACR ePrint Archive, with on average 2 or 3 highly relevant publications for us per month. With such a pace, the picture on homomorphic encryption has extensively changed since April 2015 and if we were to start our thesis today, our work would be conducted very differently. To keep only a few concurrent (and simultaneous) works, [LL15, Alb17] achieved similar and better results than what we report in Chapter 3. Then [CS16] offered similar outcomes as we did in Chapter 5. And lastly, [CGGI16, CGGI17] achieved better performances, in some cases, than we did at the same time (see Chapter 6).

Perspectives

We are definitely not at the end of our journey. Below we share our thoughts for works that could help push further the research efforts we joined during the thesis.

On targeted attacks. As most recently illustrated by the work of Albrecht [Alb17] against the settings of HElib [Hal] and SEAL [LCP17], we can still expect better attacks for such or such case. As we saw with YASHE' and F-NTRU, we shall not advocate the use of schemes that do not stand on a hard ground. Hence, when we see that today most of the homomorphic encryption libraries choose not to comply with the hardness reduction conditions, we can expect some troubles in the future.

To this end, we encourage cryptanalytic efforts or more efficient security reductions to make things clearer for everyone.

On lattice reduction. The last `fplll` days of December 2017 put a focus on implementing sieve algorithms for SVP/CVP. Indeed, we can expect that, in a near future, enumeration will be beaten by sieving for the instances of interest. Thus, lattice reduction will continue to improve, whether with generic libraries like `fplll` or with more confidential special-purpose implementation. Many heuristics remain to be experimented with and many parameters to tune. Thanks to the speed of `fplll` core and `fpylll` flexibility, we think playing within this framework should be fruitful.

On scheme improvements. Homomorphic Encryption reaches the age of maturity and we start to see industrialised framework [CL] that aims further than proof-of-concept implementations. We can expect many efficiency improvements and we think there is also space for algorithmic leaps. Also, we think there will always be several lines of candidates, each with its distinguishing features. The choice for the best will depend on the context, yet we will have generically good schemes.

Appendices

Disseminations

Talks

- Journées "Codes et Cryptographie", *Quels paramètres pour le chiffrement homomorphe sur RLWE?*, 15 octobre 2015, `http://imath.univ-tln.fr/C2`, about [BF16]
- Workshop HEAT, *Ring-LWE security in the case of FHE*, 5 July 2016, `https://wheat2016.lip6.fr`, about [BF16]
- Lattice meeting, ENS Lyon, *Large FHE gates from Tensored Homomorphic Accumulator*, 19 January 2018, about [BDF17]

Pre-prints

- *A note on Ring-LWE security in the case of fully homophic encryption*, with Caroline Fontaine [BF16]
- *Determination and exploration of practical parameters for the latest somewhat homomorphic encryption schemes*, with Vincent Migliore and Caroline Fontaine [MBF16]
- *Large FHE gates from Tensored Homomorphic Accumulator*, with Léo Ducas and Max Fillinger [BDF17]

Misc

- *Survey and analysis of DNS infrastructures*, with Damien Cémilleux, Sravani Teja Bulusu, Xiaoyang Zhu and Guillaume Valadon [**?**]

Publications

- *Homomorphic encryption: current progresses and challenges*, with Caroline Fontaine, Guy Gogniat, Vincent Herbert, Vianney Lapotre, Vincent Migliore, Adeline Roux-Langlois, Conference C2SI-Carlet 2017 [BFG+17]
- *A note on Ring-LWE in the case of Fully Homomorphic Encryption*, with Caroline Fontaine, Indocrypt 2017 [BF17]

- *Determination and exploration of practical parameters for the latest somewhat homomorphic encryption schemes*, with Vincent Migliore and Caroline Fontaine, IEEE Transactions on computers, special issue on Engineering in a Post-Quantum World, 2018 [**?**]
- *Large FHE gates from Tensored Homomorphic Accumulator*, with Léo Ducas and Max Fillinger, Africacrypt 2018 [**?**]

Source code

- Maintainer of HE8, public repository, `https://github.com/gbonnoron/Borogrove`
- Contributions to `fplll`, public repository: `https://github.com/fplll/fplll`
- Contributions to SHIELD-NFLlib, private repository: `https://github.com/gbonnoron/SHIELD-NFLlib`

List of Figures

List of Tables

List of Algorithms

Bibliography

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 16, 35, 41

[ABSS93] Sanjeev Arora, László Babai, Jacques Stern, and Z Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 724–733. IEEE, 1993. 32

[ACF+15a] Martin R Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Algebraic algorithms for LWE problems. *ACM Communications in Computer Algebra*, 49(2):62–62, 2015. 40

[ACF+15b] Martin R Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the BKW algorithm on LWE. *Designs, Codes and Cryptography*, 74(2):325–354, 2015. 40

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany. 92

[ADRS15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete Gaussian sampling: Extended abstract. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 733–742, Portland, OR, USA, June 14–17, 2015. ACM Press. 46

[ADSD15] Divesh Aggarwal, Daniel Dadush, and Noah Stephens-Davidowitz. Solving the closest vector problem in 2^n time - the discrete Gaussian strikes again! In Venkatesan Guruswami, editor, *56th FOCS*, pages 563–582, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press. 46

[AFFP14] Martin R. Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Lazy modulus switching for the BKW algorithm on LWE. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 429–445, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany. 40, 42

[AFG14] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 13*, volume 8565 of *LNCS*, pages 293–310, Seoul, Korea, November 27–29, 2014. Springer, Heidelberg, Germany. 41

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. *Automata, languages and programming*, pages 403–415, 2011. 40

[AGH10] Carlos Aguilar-Melchor, Philippe Gaborit, and Javier Herranz. Additively Homomorphic Encryption with d-Operand Multiplications. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 138–154, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 34

[AGVW17] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 297–322, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. 41

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108. ACM, 1996. 31, 34

[Ajt98] Miklós Ajtai. The shortest vector problem in L 2 is NP-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19. ACM, 1998. 31

[Alba] Martin Albrecht. fpylll, a Python wrapper from fplll library. Available at `https://github.com/fplll/fpylll`. 52

[Albb] Martin Albrecht. lwe-estimator, Sage module for estimating the concrete security of LWE instances. Available at `https://bitbucket.org/malb/lwe-estimator`. 17, 40, 79

[Albc] Martin Albrecht. lwe-generator, Sage module for generating LWE instances. Available at `https://bitbucket.org/malb/lwe-generator`. 52

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany. 15, 41, 52, 94, 109, 113, 114

[Ang17] Angela Jäschke and Frederik Armknecht. (Finite) Field Work: Choosing the Best Encoding of Numbers for FHE Computation. In *Proc. of CANS*, 2017. 84

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 36, 91

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. 14, 40, 109

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in NP ∩ coNP. *Journal of the ACM (JACM)*, 52(5):749–765, 2005. 31

[AWHT16] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 789–819, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 65

[Bab86] László Babai. On Lovász'lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. 15, 41, 47

[BDF17] Guillaume Bonnoron, Léo Ducas, and Max Fillinger. Large FHE gates from tensored homomorphic accumulator. Cryptology ePrint Archive, Report 2017/996, 2017. `http://eprint.iacr.org/2017/996`. 92, 95, 117

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th SODA*, pages 10–24, Arlington, VA, USA, January 10–12, 2016. ACM-SIAM. 62

[BEHZ16] 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 M. Heys, editors, *SAC 2016*, volume 10532 of *LNCS*, pages 423–442, St. John's, NL, Canada, August 10–12, 2016. Springer, Heidelberg, Germany. 35, 111

[BF16] Guillaume Bonnoron and Caroline Fontaine. A note on ring-LWE security in the case of fully homomorphic encryption. Cryptology ePrint Archive, Report 2016/385, 2016. `http://eprint.iacr.org/2016/385`. 117

[BF17] Guillaume Bonnoron and Caroline Fontaine. A note on ring-lwe security in the case of fully homomorphic encryption. In *International Conference in Cryptology in India*, pages 27–43. Springer, 2017. 25, 40, 52, 117

[BFG+17] Guillaume Bonnoron, Caroline Fontaine, Guy Gogniat, Vincent Herbert, Vianney Lapôtre, Vincent Migliore, and Adeline Roux-Langlois. Somewhat/fully homomorphic encryption: Implementation progresses and challenges. In Said El Hajji, Abderrahmane Nitaj, and El Mamoun Souidi, editors, *Codes, Cryptology and Information Security: Second International Conference, C2SI 2017, Rabat, Morocco, April 10–12, 2017, Proceedings - In Honor of Claude Carlet*, pages 68–82, Cham, 2017. Springer International Publishing. 117

[BFKL94] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 278–291, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany. 92

[BG14] Shi Bai and Steven D Galbraith. Lattice decoding attacks on binary LWE. In *Australasian Conference on Information Security and Privacy*, pages 322–337. Springer, 2014. 14, 41, 42

[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*, pages 309–325. ACM, 2012. 12, 35, 71

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003. 32, 40

[BLLN13] Joppe W Bos, Kristin E Lauter, Jake Loftus, and Michael Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In *IMA Int. Conf.*, pages 45–64. Springer, 2013. 12, 16, 35, 41, 71

[BM] Guillaume Bonnoron and Vincent Migliore. SHIELD-NFLlib: SHIELD implementation over NFLlib. [Private repository; online; accessed 25-October-2017]. 90

[BR15] Jean-François Biasse and Luis Ruiz. FHEW with efficient multibit bootstrapping. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 119–135, Guadalajara, Mexico, August 23–26, 2015. Springer, Heidelberg, Germany. 36

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 12, 35, 36

[BTCS+15] Ray Beaulieu, Stefan Treatman-Clark, Douglas Shors, Bryan Weeks, Jason Smith, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2015. 70

[BTS16] Shi Bai, Laarhoven Thijs, and Damien Stehlé. Tuple lattice sieving. In *ANTS 2016*, 2016. 46, 58, 62

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press. 35

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. 12, 35

[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, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. 12, 18, 36, 91, 94, 100, 103, 111, 113

[CGGI17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 377–408, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. 12, 18, 20, 92, 111, 113

[Che13] Yuanmi Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, Paris 7, 2013. 58

[CIV16] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably weak instances of ring-LWE revisited. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 147–167, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 41, 94

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. 71

[CL] CEA-LIST. Cingulata: compiler toolchain and RTE for running C++ programs over encrypted data. [Online; accessed 04-January-2018]. 114

[CLS15] Hao Chen, Kristin Lauter, and Katherine E. Stange. Attacks on search RLWE. Cryptology ePrint Archive, Report 2015/971, 2015. `http://eprint.iacr.org/2015/971`. 41

[CLS16] Hao Chen, Kristin Lauter, and Katherine E. Stange. Vulnerable galois RLWE families and improved attacks. Cryptology ePrint Archive, Report 2016/193, 2016. `http://eprint.iacr.org/2016/193`. 41

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 487–504, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. 12, 35

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany. 16, 45, 55, 58, 59, 61

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 446–464, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 12, 35

[Coh96] Henri Cohen. *A Course in Computational Algebraic Number Theory*, volume 138. Springer Berlin, 1996. 65

[Cop96] Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *Advances in cryptology—EUROCRYPT'96*, pages 178–189. Springer, 1996. 31

[Cry] CryptoExperts. FV-NFLlib: Library implementing the Fan-Vercauteren homomorphic encryption scheme. [Online; accessed 25-October-2017]. 12, 35, 90

[CS16] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 325–340, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Heidelberg, Germany. 70, 113

[CSVW16] Anamaria Costache, Nigel P. Smart, Srinivas Vivek, and Adrian Waller. Fixed-point arithmetic in SHE schemes. In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016*, volume 10532 of *LNCS*, pages 401–422, St. John's, NL, Canada, August 10–12, 2016. Springer, Heidelberg, Germany. 71

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. 12, 18, 36, 91, 103

[DS16] Yarkın Doröz and Berk Sunar. Flattening NTRU for evaluation key free homomorphic encryption. Cryptology ePrint Archive, Report 2016/315, 2016. `http://eprint.iacr.org/2016/315`. 16, 35, 41, 71

[dt17] The FPLLL development team. fplll, a lattice reduction library. Available at `https://github.com/fplll/fplll`, 2017. 15, 40, 48

[DTV15] Alexandre Duc, Florian Tramèr, and Serge Vaudenay. Better algorithms for LWE and LWR. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 173–202, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. 40

[EHL14] Kirsten Eisenträger, Sean Hallgren, and Kristin E. Lauter. Weak instances of PLWE. In Antoine Joux and Amr M. Youssef, editors, *SAC 2014*, volume 8781 of *LNCS*, pages 183–194, Montreal, QC, Canada, August 14–15, 2014. Springer, Heidelberg, Germany. 41

[ELOS15] Yara Elias, Kristin E. Lauter, Ekin Ozman, and Katherine E. Stange. Provably weak instances of ring-LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 63–92, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 41

[FJ05] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation". 105

[FP85] Ulrich Fincke and Michael Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of computation*, 44(170):463–471, 1985. 46, 58

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `http://eprint.iacr.org/2012/144`. 12, 16, 34, 35, 36, 71

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 12, 34

[GHGN06] Nicolas Gama, Nick Howgrave-Graham, and Phong Q. Nguyen. Symplectic lattice reduction and NTRU. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 233–253, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. 45

[GHPS12] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. Ring switching in BGV-style homomorphic encryption. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 19–37, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg, Germany. 95, 98

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 35

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 35, 71

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 23–42, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 40

[GMSS99] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and J-P Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999. 32

[GN08a] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 207–216, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press. 56, 58, 59

[GN08b] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. 45, 58, 61

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. 41, 46, 47, 58

[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, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 12, 35, 93

[Hal] Shai Halevi. HElib: an implementation of homomorphic encryption. [Online; accessed 25-October-2017]. 35, 94, 114

[HF17] Vincent Herbert and Caroline Fontaine. Software implementation of 2-depth pairing-based homomorphic encryption scheme. Cryptology ePrint Archive, Report 2017/091, 2017. `http://eprint.iacr.org/2017/091`. 90

[HHGP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140, San Francisco, CA, USA, April 13–17, 2003. Springer, Heidelberg, Germany. 92

[HK17] Gottfried Herold and Elena Kirshanova. Improved algorithms for the approximate k-list problem in euclidean norm. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 16–40, Amsterdam, The Netherlands, March 28–31, 2017. Springer, Heidelberg, Germany. 15, 46, 58, 62

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998. 35, 92

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 447–464, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. 58, 61

[HRP13] M Haque, Mohammad Obaidur Rahman, and Josef Pieprzyk. Analysing progressive-BKZ lattice reduction algorithm. *Proc. NCICIT*, 13:73–80, 2013. 65

[HS15] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 641–670, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. 94

[JA16] Angela Jäschke and Frederik Armknecht. Accelerating homomorphic computations on rational numbers. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 405–423, Guildford, UK, June 19–22, 2016. Springer, Heidelberg, Germany. 71

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *15th ACM STOC*, pages 193–206, Boston, MA, USA, April 25–27, 1983. ACM Press. 59

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987. 41

[KF15] Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 43–62, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 40

[KF17] Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 3–26, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany. 16, 35, 41, 71

[KGV16] Alhassan Khedr, Glenn Gulak, and Vinod Vaikuntanathan. SHIELD: scalable homomorphic implementation of encrypted data-classifiers. *IEEE Transactions on Computers*, 65(9):2848–2858, 2016. 12, 16, 35, 52, 71, 77

[KO63] A Karatsuba and Yu Ofman. Multiplication of multidigit numbers on automata. In *Soviet Physics Doklady*, volume 7, page 595, 1963. 82

[LCP17] Kim Laine, Hao Chen, and Rachel Player. Simple Encrypted Arithmetic Library - SEAL v2.3. Technical report, Microsoft Research, November 2017. 12, 15, 35, 48, 114

[Lep14] Tancrède Lepoint. A proof-of-concept implementation of the homomorphic evaluation of SIMON using FV and YASHE leveled homomorphic cryptosystems, 2014. [Online; accessed 25-October-2017]. 15, 48

[LL15] Kim Laine and Kristin Lauter. Key recovery for LWE in polynomial time. Cryptology ePrint Archive, Report 2015/176, 2015. `http://eprint.iacr.org/2015/176`. 48, 113

[LLL82] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. 14, 54, 57

[LM09] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany. 32

[LN13] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 293–309, San Francisco, CA, USA, February 25 – March 1, 2013. Springer, Heidelberg, Germany. 15, 41, 46, 48

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 14*, volume 8469 of *LNCS*, pages 318–335, Marrakesh, Morocco, May 28–30, 2014. Springer, Heidelberg, Germany. 48, 70, 77, 79, 90

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339, San Francisco, CA, USA, February 14–18, 2011. Springer, Heidelberg, Germany. 41, 47, 51, 58, 70

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. 14, 33, 41, 94

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015. 33

[MBF16] Vincent Migliore, Guillaume Bonnoron, and Caroline Fontaine. Determination and exploration of practical parameters for the latest Somewhat Homomorphic Encryption (SHE) Schemes. working paper or preprint, October 2016. 117

[Mic01] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM journal on Computing*, 30(6):2008–2035, 2001. 31

[MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In Leonard J. Schulman, editor, *42nd ACM STOC*, pages 351–358, Cambridge, MA, USA, June 5–8, 2010. ACM Press. 46, 58

[MW16] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 820–849, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 45, 55, 56, 58, 59

[NS05] Phong Q. Nguyen and Damien Stehlé. Floating-point LLL revisited. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 215–233, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. 58

[NS06] Phong Nguyen and Damien Stehlé. LLL on the average. *Algorithmic Number Theory*, pages 238–256, 2006. 58

[NS09] Phong Q Nguyen and Damien Stehlé. An LLL algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009. 58, 59

[NV09] Phong Q Nguyen and Brigitte Vallée. *The LLL Algorithm: Survey and Applications*. Springer Science & Business Media, 2009. 58

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 34

[Pei16] Chris Peikert. How (not) to instantiate ring-LWE. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 411–430, Amalfi, Italy, August 31 – September 2, 2016. Springer, Heidelberg, Germany. 14, 41, 88, 94

[Puj08] Xavier Pujol. Recherche efficace de vecteur court dans un réseau euclidien. *Mémoire de M2*, 2008. 16, 59

[Qua] Quarkslab and CryptoExperts and INP ENSEEIHT. NFLlib: An NTT-based Fast Lattice library. [Online; accessed 25-October-2017]. 90

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978. 12, 34

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 13, 32, 79, 92

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 34

[RSSS17] Miruna Rosca, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Middle-product learning with errors. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 283–297, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 34

[Sch03] Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. In *STACS*, volume 2607, pages 145–156. Springer, 2003. 56

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994. 14, 55, 59, 61

[Sho] Vitor Shoup. NTL – A Library for doing Number Theory. [Online; accessed 25-October-2017]. 48

[SS11] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 27–47, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany. 41

[SV14] Nigel P Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Designs, codes and cryptography*, pages 1–25, 2014. 35, 84

[Too63] Andrei L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics Doklady*, volume 3, pages 714–716, 1963. 82

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. 12, 35

[vdPS13] Joop van de Pol and Nigel P. Smart. Estimating key sizes for high dimensional lattice-based systems. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 290–303, Oxford, UK, December 17–19, 2013. Springer, Heidelberg, Germany. 58, 70

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : En route vers un chiffrement complètement homomorphe applicable

Mots clés : Chiffrement, Cloud, Homomorphe, Post-Quantique, Sécurité

Résumé : Craig Gentry a proposé en 2009 le premier schéma de chiffrement complétement homomorphe. Depuis, un effort conséquent a été, et est toujours, fourni par la communauté scientifique pour rendre utilisable ce nouveau type de cryptographie. Son côté révolutionnaire tient au fait qu'il permet d'effectuer des traitements directement sur des données chiffrées (sans que l'entité réalisant les traitements ait besoin de les déchiffrer). Plusieurs pistes se sont développées en parallèle, explorant d'un côté des schémas complétement homomorphes, plus flexibles en termes d'applications mais plus contraignants en termes de taille de données ou en coût de calcul, et de l'autre côté des schémas quelque peu homomorphes, moins flexibles mais aussi moins coûteux.

Cette thèse, réalisée au sein de la chaire de cyberdéfense des systèmes navals, s'inscrit dans cette dynamique. Nous avons endossé divers rôles. Tout d'abord un rôle d'attaquant pour éprouver la sécurité des hypothèses sous-jacentes aux propositions. Ensuite, nous avons effectué un état de l'art comparatif des schémas quelque peu homomorphes les plus prometteurs afin d'identifier le(s) meilleur(s) selon les cas d'usages, et de donner des conseils dans le choix des paramètres influant sur leur niveau de sécurité, la taille des données chiffrées et le coût algorithmique des calculs. Enfin, nous avons endossé le rôle du concepteur en proposant un nouveau schéma complétement homomorphe performant, ainsi que son implémentation mise à disposition sur github.

Title : A journey towards practical Fully Homomorphic Encryption

Keywords : Encryption, Cloud, Homomorphic, Post-Quantum, Security

Abstract : Craig Gentry presented in 2009 the first fully homomorphic encryption scheme. Since then, a tremendous effort has been, and still is, dedicated by the cryptographic community to make practical this new kind of cryptography. It is revolutionnary because it enables direct computation on encrypted data (without the need for the computing entity to decrypt them). Several trends have been developed in parallel, exploring on one side fully homomorphic encryption schemes, more versatile for applications but more costly in terms of time and memory. On the other side, the somewhat homomorphic encryption schemes are less flexible but more efficient.

This thesis, achieved within the Chair of Naval Cyber Defence, contributes to these trends. We have endorsed different roles. First, an attacker position to assess the hardness of the security assumptions of the proposals. Then, we conducted a state-of-the-art of the most promising schemes in order to identify the best(s) depending on the use-cases and to give precise advice to appropriately set the parameters that drive security level, ciphertext sizes and computation costs. Last, we endorsed a designer role. We proposed a new powerful fully homomorphic encryption scheme together with its open-source implementation, available on github.