Implantation matérielle de chiffrements homomorphiques

Asma Mkhinini

Communauté UNIVERSITÉ Grenoble Alpes

UNIVERSITE DE SOUSSE

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

préparée dans le cadre d'une cotutelle *entre la Communauté Université Grenoble Alpes* et l'université de Sousse

Spécialité : **Nanoélectronique et Nanotechnologies**

Arrêté ministériel : 7 août 2006

Présentée par

Asma MKHININI

Thèse dirigée par **Régis LEVEUGLE**
Co-dirigée par **Rached TOURKI**
Co-encadrée par **Paolo MAISTRI**

préparée au sein du **Laboratoire TIMA**
dans **l'École Doctorale EEATS**

et du **Laboratoire EμE**
dans **L'Ecole Nationale d'Ingénieurs de Sousse**

Implantation matérielle de chiffrements homomorphiques

Thèse soutenue publiquement le 14 décembre 2017,
devant le jury composé de :

M. Renaud SIRDEY
Directeur de recherche, CEA, Président
M. Guy GOGNIAT
Professeur, Université de Bretagne-Sud, Rapporteur
M. Mohamed HAMDI
Maître de conférences, Université de Carthage, Rapporteur
M. Régis LEVEUGLE
Professeur, Université Grenoble Alpes, Grenoble-INP, Directeur de thèse
M. Rached TOURKI
Professeur émérite, Université de Monastir, Co-directeur de thèse
M. Paolo MAISTRI
Chargé de recherche, CNRS, Co-encadrant de thèse

Remerciements

Table des matières

Liste des figures

Liste des tableaux

Annexe A. Exemples de paramètres pratiques pour des schémas homomorphes basés sur RLWE

Introduction générale

La cryptographie est aujourd'hui omniprésente dans notre vie quotidienne. Elle se trouve au sein de nos téléphones, de nos ordinateurs, de nos cartes bancaires, afin de protéger nos données personnelles et préserver notre vie privée. Les premières tentatives de cryptographie remontent à l'antiquité et avaient principalement comme objectif de garantir la confidentialité des messages. Ce but était atteint grâce à des jeux de substitutions et de permutations. Puis la cryptographie a connu au fur et à mesure des années plusieurs pivots dont l'un des plus importants est dû aux défis soulevés par la numérisation. En effet, il faut, en plus de la confidentialité, garantir l'authenticité et l'intégrité des échanges. Le problème n'apparaissait pas de manière aussi importante lorsque les communications se faisaient sur support physique. L'authenticité d'un message est garantie par une signature. Or, contrefaire une signature écrite reste plus compliqué que de copier-coller une signature numérique. De même, modifier, de manière imperceptible, un message écrit sur du papier est une tâche bien plus difficile que d'effacer quelques octets d'informations dans une transmission numérique. Ainsi, en plus du chiffrement, la communauté cryptographique a développé de nouveaux outils pour répondre à ces questions nées du numérique.

Les nombreux standards de chiffrement, de signature et d'authentification de messages, tous approuvés par la communauté scientifique, pourraient donner l'impression que la cryptographie a parfaitement réussi la transition entre la phase artisanale et la phase numérique. Néanmoins, de nouveaux enjeux liés à la transmission de l'information émergent constamment. La question n'est plus de transmettre à grande échelle, mais aussi d'exploiter cette masse d'information disponible sur internet afin de la rendre significative.
L'analyse de masses de données est devenue une pratique omniprésente dans ce qu'il est convenu d'appeler le monde connecté. Elle est censée permettre de prévoir la météo, les comportements des consommateurs, les fluctuations des marchés, les analyses médicales, … La place de ces analyses dans notre quotidien, et surtout dans le quotidien des décideurs du monde économique, vient du fait qu'elle permet de réduire l'incertitude liée au processus de décision : elle établit des choix optimaux en termes de confort et de profit. Par exemple, un fournisseur de stockage externe a intérêt à vérifier que les données stockées par les utilisateurs ne contiennent pas de duplicata afin de maintenir les coûts de la plateforme bas. D'autres services sont intrinsèquement appelés à inspecter les données des utilisateurs. Par exemple, un fournisseur de logiciel de contrôle parental doit, pour assurer la sécurité des plus jeunes, analyser le trafic internet émis et reçu par un ordinateur. Les utilisateurs profitent également de l'exploitation de leurs données personnelles. Par exemple, la géo-localisation des mobiles permet à des fournisseurs de services GPS de calculer le chemin le plus rapide en fonction du trafic routier en temps réel.

La prise de conscience autour de l'utilisation de nos données personnelles par les acteurs d'internet est croissante. En effet, nos données de navigation ou d'utilisation de nos

smartphones révèlent une grande quantité d'information sur nous : notre lieu de domicile, de travail, nos trajets quotidiens, notre condition médicale. Une première solution à ce problème vient d'algorithmes d'anonymisation, dont le but est de préserver la valeur monétaire des informations collectées auprès des utilisateurs, sans révéler leur vie privée. Cependant, l'anonymisation ne répond qu'à la question de l'exploitation de données agrégées.
Pour des données individuelles, comme le trafic émis et reçu par un seul ordinateur, seul le chiffrement apparaît comme une réponse adaptée pour en préserver la confidentialité. Or, chiffrer ces données empêche leur analyse. Le chiffrement priverait alors les utilisateurs d'un grand nombre de services auxquels ils sont désormais habitués. Les enjeux de la cryptographie se transforment ainsi au fur et à mesure de l'évolution des communications. Il ne suffit plus d'assurer la confidentialité, l'intégrité et l'authenticité d'un message. Il faudrait aussi pouvoir exploiter les données chiffrées afin de maintenir un niveau de service et d'ergonomie, tout en respectant la vie privée des utilisateurs.

Dans ce contexte, l'idéal serait de pouvoir calculer sur des données chiffrées sans rien apprendre des données claires sous-jacentes. Cette possibilité ouvre un monde quasiment magique, où il n'est plus nécessaire de voir l'information pour l'analyser. Cela permettrait de déléguer des calculs sur des données sensibles, sans faire confiance à la structure opérant ces calculs.
Une solution permettant de réaliser les traitements dans le domaine chiffré est le chiffrement homomorphe. Cet adjectif est dérivé du grec et signifie "même forme". Introduit pour la première fois en 1978, ce concept est resté un problème ouvert pendant une trentaine d'années. De nombreux systèmes de chiffrement permettent en effet d'effectuer soit des additions, soit des multiplications sur les chiffrés, mais un protocole permettant un enchainement arbitraire de ces deux opérations n'est proposé qu'en 2009, par Craig Gentry.
Les premières instanciations de cette surprenante primitive ne peuvent être considérées comme pratiques ; chiffrer un bit de données engendre des données très grandes qui voient leur taille augmenter au fur et à mesure des calculs et chaque multiplication de deux bits chiffrés nécessite d'être suivie par une procédure de plusieurs dizaines de minutes.

Nous nous intéressons dans ce manuscrit aux solutions permettant d'améliorer l'efficacité de la cryptographie homomorphe dans un contexte d'implantation matérielle.

Ce manuscrit est divisé en 4 chapitres. Je commence dans le chapitre I par aborder les outils mathématiques et cryptographiques sous-jacents au chiffrement homomorphe. Je présente, ensuite, les différents schémas introduits dans l'état de l'art depuis la percée de Gentry. Cette étude théorique vise essentiellement à montrer l'évolution significative de ce domaine de recherche et à comparer l'efficacité des cryptosystèmes.
Le chapitre II est consacré à l'aspect pratique du chiffrement homomorphe et décrit les implantations existantes dans l'état de l'art. Ceci nous permettra d'évaluer les performances des différentes propositions et d'identifier les pistes à explorer.
Le chapitre III propose un accélérateur matériel de la fonction la plus coûteuse et la plus fréquente dans la construction des schémas homomorphes les plus prometteurs. Cette fonction

est la multiplication polynomiale modulaire dans un anneau de polynômes. Je propose une nouvelle approche en abordant ce sujet sur la base d'une décomposition des opérandes et d'une exploitation de la synthèse de haut niveau. Les accélérateurs proposés ont la particularité d'être très flexibles et peuvent supporter des opérandes très grands de taille arbitraire.
Dans le chapitre IV, je m'intéresse à une deuxième fonction qui est l'échantillonne gaussien et je présente une implantation matérielle de cette primitive. Les deux accélérateurs proposées sont utilisés afin d'évaluer l'un des cryptosystèmes homomorphes les plus récents et les plus efficaces.

Chapitre I. Chiffrement homomorphique

Ce chapitre dresse un état de l'art des schémas de chiffrement homomorphique qui a connu une grande évolution depuis l'introduction du concept par Rivest, Adleman et Dertouzos [RAD78], il y a une quarantaine d'années. Nous commençons par introduire les notions mathématiques et cryptographiques nécessaires à la compréhension de la suite de ce manuscrit. Nous donnerons ensuite un aperçu général sur les schémas partiellement homomorphes avant de décrire le premier système complètement homomorphe inventé par Craig Gentry [Gen09a] puis. nous nous focaliserons sur les schémas qui ont suivi cette percée théorique. Enfin, nous conclurons ce chapitre avec les applications existantes et envisagées de la cryptographie homomorphe.

I.1. Préliminaires

I.1.1. Définitions

Nous commençons par introduire quelques notions relatives à la théorie de la complexité. Pour plus de détails, nous renvoyons le lecteur à [AB09]. Le but de cette théorie est de classifier les problèmes algorithmiques en fonction de leur difficulté. Nous nous intéressons à la complexité en temps qui mesure le nombre d'opérations, en fonction de la taille des entrées d'un problème, pour qu'un algorithme puisse le résoudre. Pour l'évaluation des algorithmes, notre modèle de calcul sera la machine de Turing. Il s'agit d'une machine à états finis avec une mémoire en écriture et en lecture sous la forme d'un ruban infini.

Problème de décision :
Un problème désigne la description générale d'une question comme, par exemple, trouver un facteur non trivial d'un entier (s'il existe). Quand on se donne une valeur particulière de l'entier en question, on a une instance du problème. Pour résoudre un tel problème on essaie de décrire un algorithme*.
Les problèmes de décision sont ceux qui consistent à répondre par oui ou par non à une question donnée.

Notation de Landau :
Pour mesurer la complexité d'un algorithme résolvant un problème, nous utiliserons la notation classique de Landau "grand O ", définie de la manière suivante :

on écrira $f(n) = \mathcal{O}(g(n))$ s'il existe une constante $c > 0$ et un entier $n_0 > 0$, tels que $f(n) \leq c.g(n)$ pour tout $n > n_0$

Problème linéaire, problème polynomial, problème exponentiel :
Soit n la taille de l'entrée d'un problème algorithmique.
- Le problème est linéaire si la complexité du meilleur algorithme connu le résolvant est $\mathcal{O}(n)$.

* C'est cette notion que l'on pense formaliser au mieux grâce à une machine de Turing.

- Le problème est polynomial s'il existe un polynôme P tel que la complexité du meilleur algorithme connu le résolvant est $\mathcal{O}(P(n))$.
- Le problème est exponentiel si la complexité du meilleur algorithme connu le résolvant est $\mathcal{O}(2^n)$.

Classe P, Classe NP :
Un problème de décision D est dans la classe P (resp. NP) si une machine de Turing déterministe (resp. non déterministe) décide D en temps polynomial par rapport à la taille de l'entrée.
Remarque 1 : Une machine de Turing non déterministe est une variante de la machine de Turing normale qui devine les solutions. La machine devine une solution d'un problème, soit en faisant par chance une bonne hypothèse, soit en essayant toutes les possibilités en parallèle. Elle vérifie son hypothèse en un temps polynomial. Ainsi, la vérification du problème se fait en un temps polynomial, mais le fait d'essayer toutes les possibilités apporte un nombre de cas à explorer très grand, ce qui rend l'algorithme inefficace.

Réduction d'un problème :
Afin de déterminer la difficulté d'un problème, on peut la comparer à celle d'un autre problème au moyen d'une réduction. La réduction est une transformation d'un problème P1 dont la complexité est connue en un problème P2 dont la complexité est inconnue. Sous certaines conditions, elle permet de montrer que résoudre P2 est au moins aussi difficile que résoudre P1.
Définition I.1. En résolvant un problème P1, nous dirons que nous faisons appel à un P2-oracle si l'algorithme que nous utilisions crée à partir de l'instance initiale du problème P1, une instance du problème P2.
Définition I.2. Soient P1 et P2 deux problèmes. Nous dirons que P1 se réduit à P2 (en temps polynomial) s'il existe un algorithme en temps polynomial pour P1 qui utilise au plus un nombre polynomial d'appels à un P2-oracle.

NP-difficile et NP-complet :
Définition I.3. Un problème Q est NP-difficile si tout problème de la classe NP est réductible à Q. Si de plus Q est dans la classe NP, nous dirons qu'il est NP-complet.

I.1.2. Rappels mathématiques

I.1.2.1. Groupe

Définition I.4. On appelle groupe tout ensemble non-vide $\mathbb{G}$ muni d'une loi de composition interne $\star$, vérifiant les 3 propriétés suivantes :

(a) la loi $\star$ est associative dans $\mathbb{G}$; $\forall\,(x,y,z) \in G^3\,, x \star (y \star z) = (x \star y) \star z$
(b) la loi $\star$ admet un élément neutre dans $\mathbb{G}$; $\exists\, e\ \mathbb{G}$ tel que $\forall\, x \in\ \mathbb{G},\ x \star e = e \star x = x$
(c) tout élément de $\mathbb{G}$ admet un symétrique dans $\mathbb{G}$ pour la loi $\star$; $\forall\, x \in\ \mathbb{G}$, il existe un élément $y \in\ \mathbb{G}$ tel que $x \star y = y \star x = e$

Définition I.5. On appelle groupe commutatif, ou groupe abélien, tout groupe $\mathbb{G}$ dont la loi $\star$ vérifie de plus la condition supplémentaire de commutativité: $x \star y = y \star x$ pour tous $x, y \in\ \mathbb{G}$.

Définition I.6. On appelle groupe fini un groupe $\mathbb{G}$ qui, en tant qu'ensemble, n'a qu'un nombre fini d'éléments. Ce nombre d'éléments (qui n'est autre que le cardinal de l'ensemble $\mathbb{G}$) est appelé l'ordre du groupe $\mathbb{G}$, noté $|\mathbb{G}|$.

Définition I.7. Soit $(\mathbb{G},\star)$ un groupe.
- Sous-groupe engendré par un élément : soit $x \in \mathbb{G}$, l'ensemble $\{x^n, x \in \mathbb{N}\}$ que l'on notera $\langle x \rangle$ est également un groupe que l'on appelle sous-groupe engendré par x.
- Groupe cyclique : un groupe $\mathbb{G}$ est cyclique s'il existe $x \in \mathbb{G}$ tel que $\langle x \rangle = \mathbb{G}$. Un tel élément sera appelé générateur de $\mathbb{G}$.

Les groupes les plus utilisés en cryptographie ont la particularité d'être d'ordre premier, donc cycliques (théorème de Lagrange sur les groupes [Bou07]), et tous les éléments du groupe différents de l'élément neutre sont des générateurs du groupe.

Exemples :
- L'ensemble $\mathbb{Z}$ muni de l'addition est un groupe.
- $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z} = \{\bar{0}, \bar{1}, \dots, \overline{n-1}\}$ l'ensemble des classes d'équivalence modulo n, muni de l'addition est un groupe.
  On appelle classe d'équivalence de x modulo n l'ensemble des entiers congrus à x modulo n. On note $\bar{x}$ la classe de x.
- $\mathbb{Z}_p^\times = (\mathbb{Z}/p\mathbb{Z})^\times$ est un groupe pour la multiplication.
- Les deux groupes $(\mathbb{Z}_p, +)$ et $(\mathbb{Z}_p^\times, \times)$ sont souvent utilisés en cryptographie (RSA et ElGamal par exemple).

Définition I.8. Soient $\mathbb{G}$ et $\mathbb{G}'$ deux groupes. Une application $f: \mathbb{G} \to \mathbb{G}'$ est un morphisme de groupes si :

$$f(x.y) = f(x).f(y) \text{ pour tous } x, y \in \mathbb{G}$$

Si f est bijective et f^{-1} est aussi un morphisme, on dit que f est un isomorphisme, si de plus $\mathbb{G} = \mathbb{G}'$, on dit que f est un automorphisme.
On dit parfois homomorphisme pour un morphisme de groupes.

I.1.2.2. Anneau

Définition I.9. Un anneau est un ensemble A muni de deux lois de composition interne +, $*$ telles que :

(i) $(A, +)$ est un groupe abélien ;
(ii) la loi $\star$ est associative, munie d'un élément neutre 1;
(iii) la loi $\star$ est distributive par rapport à la loi +, c'est-à-dire, $a \star (b + c) = a \star b + a \star c$ et $(b + c) \star a = b \star a + c \star a$ pour tous $a, b, c \in A$.

Si la loi $\star$ est commutative, on dit que A est un anneau commutatif.
Exemples :
- $(\mathbb{Z}, +, \times)$ est un anneau qu'on appelle l'anneau des entiers relatifs.
- $(\mathbb{Z}/n\mathbb{Z}, +, \times)$
- si A est un anneau (commutatif), on a aussi l'anneau des polynômes $A[x_1, x_2, \dots, x_n]$ à coefficients dans A

Définition I.10. Un corps est un anneau non nul dans lequel tout élément non nul admet un inverse pour la loi de multiplication $\times$.
Exemple : $\mathbb{Z}/p\mathbb{Z}$ où p est premier est un corps.
Définition I.11. Un morphisme (ou un homomorphisme d'anneaux) est une application $f: A \to B$ telle que :

- $f(1) = 1$
- $f(x + y) = f(x) + f(y)$
- $f(xy) = f(x)f(y)$

Définition I.12. Une partie $I \subset A$ est un idéal de A si I est un sous-groupe de A pour l'addition et si, pour tout $x \in I$ et tout $a \in A$, on a $ax \in I$.

I.1.2.3. Réseau euclidien

Intuitivement, un réseau euclidien est un ensemble de points régulièrement espacés dans un espace euclidien $\mathbb{R}^n$. L'exemple le plus simple est $\mathbb{Z}^n$ formé par tous les points avec des coefficients entiers. Tous les sous groupes de $\mathbb{Z}^n$ sont aussi des réseaux.

Définition I.13. Un réseau euclidien est un ensemble de la forme $\mathcal{L} = \sum \mathbb{Z} b_i \subseteq \mathbb{R}^n$ avec des vecteurs b_i linéairement indépendants. Ces vecteurs forment une base du réseau. Celui-ci en possède une infinité (quand la dimension est supérieure à 2), qui sont liées entre elles par des transformations uni-modulaires (c'est-à-dire linéaires, inversibles et à coefficients entiers).
Toutes les bases d'un réseau $\mathcal{L}$ ont le même nombre d'éléments appelés rang ou dimension du réseau.
La Figure I.1 illustre un exemple de réseau de dimension 2, avec deux de ses bases.

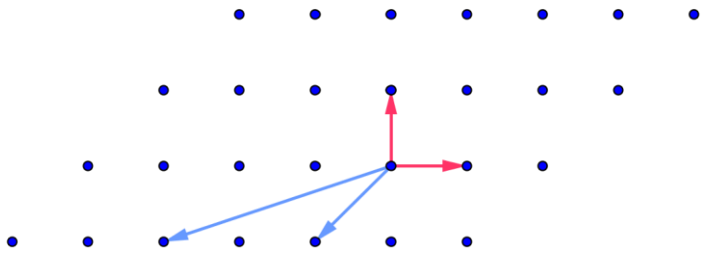

Figure I.1. Réseau euclidien de dimension 2 avec deux bases

I.1.3. Outils cryptographiques

La cryptographie, étymologiquement "écriture secrète", est l'art de rendre inintelligible un message pour ceux qui ne sont pas habilités à en prendre connaissance.
On distingue la cryptographie symétrique (ou à clé secrète) et la cryptographie asymétrique (ou à clé publique). La cryptographie symétrique est fondée sur la connaissance mutuelle par les deux interlocuteurs d'une même clef secrète. La cryptographie asymétrique ne requiert pas de secret partagé entre les deux interlocuteurs ; celui qui reçoit le message possède deux clés, une

publique qu'il diffuse, et une autre qu'il garde secrète. L'objet numérique appelé clé publique permet de chiffrer un message tandis que la clé privée associée permet de le déchiffrer.
Chacun de ces schémas fait appel à trois algorithmes : génération de clés, chiffrement et déchiffrement.

I.1.3.1. Chiffrement homomorphique

Le chiffrement homomorphique ou homomorphe, dérivé du grec et signifiant "même forme", permet de transposer la structure des messages clairs dans l'espace des chiffrés. Ainsi, une opération sur les chiffrés se traduira par une opération sur les clairs.
De manière plus formelle, un tel schéma est défini avec la syntaxe suivante :

- Algorithme de génération de clés $(\boldsymbol{pk}, \boldsymbol{sk}) \leftarrow \mathbf{KeyGen}(\mathbf{1}^{\boldsymbol{\lambda}})$: prend en entrée un paramètre de sécurité λ et retourne une clé publique pk et une clé privée sk *.
- Algorithme de chiffrement $\boldsymbol{c} \leftarrow \mathbf{Enc}(\boldsymbol{pk}, \boldsymbol{m})$: prend en entrée la clé publique pk et un message m et retourne un chiffré c.
- Algorithme de déchiffrement $\boldsymbol{m} \leftarrow \mathbf{Dec}(\boldsymbol{sk}, \boldsymbol{c})$: prend en entrée la clé privée sk et un chiffré c et retourne un message m.

Il rajoute, par rapport aux schémas classiques un :

- Algorithme d'évaluation $\boldsymbol{c} \leftarrow \boldsymbol{Eval}\ (\boldsymbol{pk}, \boldsymbol{f}, < \boldsymbol{c_1}, \dots, \boldsymbol{c_t} >)$: prend en entrée la clé publique pk, une fonction f et un vecteur de chiffrés $(c_1, \dots, c_t)$ chiffrant respectivement les messages $(m_1, \dots, m_t)$. Il retourne un chiffré $c = Enc(pk, f(m_1, \dots, m_t))$.

L'ensemble $\mathcal{F}$ des fonctions f que peut accepter le schéma comme entrée à Eval tout en garantissant un déchiffrement correct** définit ses capacités homomorphes. Cet ensemble est défini en fonction des paramètres du schéma et peut être exprimé à travers plusieurs modèles de calcul : modèle des machines RAM (Random Access Machine) [GHR+14], polynômes de degré spécifique, circuits (booléen ou plus généralement arithmétique), etc.
Quand on parlera de modèle de calcul dans la suite, on s'intéressera soit aux circuits soit aux polynômes. Le but n'étant pas de présenter la théorie des modèles de calcul, cette partie ne sera pas plus détaillée, mais on peut affirmer que la classe des algorithmes qui peuvent s'exprimer par un polynôme ou un circuit est très grande [Sav97]. Ainsi, si l'on veut exécuter un algorithme (un programme ou une fonction) sur les données chiffrées, il suffit de pouvoir transformer l'algorithme en polynôme/circuit et de choisir le chiffrement homomorphe permettant d'évaluer ce polynôme/circuit.

Circuit arithmétique :
Dans [Gen14], Gentry a montré que les calculs traversant la couche de chiffrement peuvent être modélisés sous la forme de circuits arithmétiques. Ce modèle peut se simuler par une machine de Turing. Et toute machine de Turing à n étapes peut être représentée avec un circuit de taille $\mathcal{O}(n \log(n))$.

* Dans ce manuscrit, on s'intéressera aux cryptosystèmes homomorphes à clé publique. Mais la majorité des schémas peuvent être symétrique ou asymétrique. L'auteur de [Rot11] montre comment passer d'une instanciation symétrique vers une instanciation asymétrique du chiffrement complètement homomorphe (FHE).
** L'inégalité $Dec(sk, c) \neq f(m_1, \dots, m_t)$ n'arrive qu'avec une probabilité négligeable.

Un circuit arithmétique est un ensemble de portes additives (calculent la somme de leurs entrées), portes multiplicatives (calculent le produit de leurs entrées) et portes de multiplication scalaire. Toutes ces opérations sont effectuées dans un anneau. Quand on travaille dans $\mathbb{F}_2$ et si chaque porte possède au plus 2 entrées, on parle de circuit booléen.

Définition I.14. Soit C un circuit arithmétique et soit P un chemin dans C. La profondeur multiplicative de C est le nombre maximal de portes multiplicatives présentes sur un chemin pris sur l'ensemble des chemins de C.

I.1.3.2. Sécurité prouvée

En cryptographie, il est important de s'assurer de la sécurité des schémas. Nous utilisons pour cela des preuves de sécurité établies dans des modèles de sécurité.
Nous avons abordé dans la section I.1.1 la notion de la difficulté à résoudre certains problèmes. Ce qui nous intéresse maintenant est de déterminer la difficulté de casser un schéma. Pour prouver la sécurité, on effectue une réduction d'un problème algorithmique à un problème de sécurité. Nous partons d'un problème connu pour être difficile, et prouvons qu'obtenir certaines informations sur les chiffrés avec le protocole évalué permet de résoudre notre problème de départ.
Dans les années 80, les chercheurs ont formellement défini les notions de sécurité. Goldwasser et Micali [GM84] ont introduit la notion de sécurité sémantique. Cette notion est très forte ; elle couvre l'exigence de ne pas pouvoir extraire une information, même partielle sur le clair à partir du chiffré. Bien que la sécurité sémantique soit la propriété désirée pour tout schéma de chiffrement, on étudie souvent la notion de l'indistinguabilité (IND), plus simple à analyser. Il a été démontré que ces deux notions sont équivalentes [GM84] [MRS88]. La propriété d'indistinguabilité exige que le schéma de chiffrement soit probabiliste (ajoute de l'aléa). L'attaquant tente de distinguer les chiffrés de deux messages différents de son choix. Pour ce faire, il peut avoir différentes ressources à sa disposition.

- L'attaque à clairs choisis CPA (Chosen Plaintexts Attack) : l'adversaire peut obtenir les chiffrés des messages de son choix.
- L'attaque à chiffrés choisis CCA (Chosen Ciphertexts Attack) : l'adversaire peut obtenir, pendant une période précise de son attaque, le déchiffrement des chiffrés de son choix.
- L'attaque à chiffrés choisis adaptative CCA2 : l'adversaire peut obtenir tout au long de son attaque le déchiffrement des chiffrés de son choix. Il peut donc adapter ses choix au cours de l'attaque.

On dénote un schéma qui est sûr au sens XXX (par exemple, IND) face aux attaques YYY (par exemple, CCA) un schéma XXX-YYY sûr (par exemple, IND-CCA sûr).

I.1.3.3. Problèmes classiques et hypothèses standards

Rappelons certaines des hypothèses algorithmiques sur lesquelles reposent les primitives habituelles.

Factorisation :
Soit n un module RSA, c'est-à-dire de la forme $n = pq$ avec p et q premiers. Le problème de la factorisation consiste à retrouver p et q à partir de n.

Problème RSA :
Soit n un module RSA, $y \in \mathbb{Z}_n^*$ et $e \geq 3$ premier avec l'indicatrice d'Euler $\varphi(n)$. Le problème RSA consiste, à partir de y, en connaissant n et e, à trouver sa racine e-ième, c'est-à-dire $x \in \mathbb{Z}_n^*$ tel que $y \equiv x^e \ mod n$.

Hypothèse décisionnelle de Diffie-Hellman DDH :
L'hypothèse DDH signifie que quelques soient $x, y, z \in \mathbb{Z}_q^*$, il est calculatoirement difficile de distinguer les deux triplets (g^x, g^y, g^{xy}) et (g^x, g^y, g^z).

I.1.3.4. Problèmes sous jacents à la cryptographie homomorphe

La sécurité des schémas cryptographiques est souvent ramenée à la difficulté de résolution d'un problème mathématique dit "difficile". Je me focalise ici sur quelques problèmes sous-jacents à la cryptographie basée sur les réseaux euclidiens car ils sont les plus utilisés dans ce manuscrit. Les autres problèmes et notions de sécurité seront définis quand ils seront évoqués.

Les réseaux euclidiens ont été utilisés en cryptanalyse avant de devenir un outil très puissant de conception de schémas cryptographiques grâce aux travaux d'Ajtai [Ajt96] et de Regev [Reg05] [Reg09]. La cryptographie basée sur les réseaux est considérée aujourd'hui comme l'alternative la plus plausible à la cryptographie traditionnelle reposant sur la factorisation de grands entiers et le logarithme discret. Elle possède des propriétés très intéressantes : une grande flexibilité, une sécurité basée sur des problèmes dans le pire cas, une complexité asymptotique pouvant être quasi-optimale, etc. [LLS14]. Elle est la base de plusieurs schémas de chiffrement homomorphe étudiés dans nos travaux.

Le problème du vecteur le plus court SVP (The Shortest Vector Problem):
Etant donnée une base d'un réseau, résoudre le SVP consiste à trouver un plus court vecteur non nul dans le réseau. La norme de ce vecteur est appelée minimum du réseau. Cette définition dépend de la norme dont est muni $\mathbb{R}^n$.

Le problème du vecteur le plus proche CVP (The Closest Vector Problem):
Etant donné une base et un vecteur cible t, le but est de trouver un vecteur du réseau le plus proche possible de t.

Le problème du décodage borné BDD (Bounded Distance Decoding):
Il s'agit d'une instance particulière du problème CVP.

Définition I.15. Bounded Distance Decoding BDDα : Soit $\alpha > 0$. Etant donné une base B d'un réseau $\mathcal{L}$ et un vecteur cible t tel que $dist(t, B) < \alpha\lambda_1(B)$, avec $\lambda_1(B)$ le minimum du réseau, le but est de trouver le vecteur le plus proche possible de t.

Le problème SIS (Short Integer Solution):
Le problème SIS consiste à trouver une solution non triviale à une équation linéaire modulo un entier, de sorte que les coefficients de cette solution ne soient pas trop grands.

Le problème LWE (Learning With Errors): [Reg05][Reg09]
Informellement, le problème LWE consiste à résoudre un système linéaire surdéterminé mais bruité, modulo un entier q. Le problème LWE est défini autour de trois paramètres : la dimension n, le module q et le facteur d'erreur α. Les paramètres q et α sont choisis en fonction de n. On distingue deux versions du problème LWE. Sa version calculatoire consiste à retrouver un vecteur $s \in \mathbb{Z}_q^n$ à partir d'un nombre arbitraire de produits scalaires bruités $\langle s, a_i \rangle$. Les a_i sont connus et choisis uniformément dans $\mathbb{Z}_q^n$. La version décisionnelle consiste à distinguer ces produits scalaires bruités de la distribution uniforme. [LLS14]

Définition I.16. Soit $n \geq 1$, $q \geq 2$ deux entiers, et $\alpha \in]0,1[$. La distribution $D_{n,q,\alpha}^{LWE}(s)$, pour $s \in \mathbb{Z}_q^n$, est la distribution sur $\mathbb{Z}_q^{n+1}$ obtenue par l'expérience suivante :

$$a \hookleftarrow U(\mathbb{Z}_q^n)\ ;\ e \hookleftarrow D_{\mathbb{Z},\alpha q}\ ;\ \text{renvoyer } (a, \langle a, s \rangle + e) \in \mathbb{Z}_q^{n+1}$$

Définition I.17.
Version calculatoire de LWE de paramètres n, q et α : soit $s \in \mathbb{Z}_q^n$; étant donné un nombre arbitraire d'échantillons de $D_{n,q,\alpha}^{LWE}(s)$, trouver s.
Version décisionnelle de LWE de paramètres n, q et α : soit $s \hookleftarrow U(\mathbb{Z}_q^n)$; distinguer entre les distributions $D_{n,q,\alpha}^{LWE}(s)$ et $(\mathbb{Z}_q^{n+1})$.
Ces deux versions se réduisent l'une à l'autre en temps polynomial.

Le problème RLWE (Ring Learning With Errors):
En 2010, Lyubashevsky et al. [LPR13] ont introduit une variante du LWE faisant intervenir des anneaux de polynômes de la forme $R_q = \mathbb{Z}_q/f(x)$ avec $f(x)$ un polynôme irréductible*. Nous ne décrivons pas cette variante car elle est très similaire au LWE à l'exception du fait qu'elle est définie dans l'anneau R_q et que a et s sont des polynômes et plus des vecteurs.
Plus de détails sur RLWE sont mentionnés à [LPR13].

Nous détaillerons des aspects plus pratiques des problèmes LWE et RLWE et verrons ce que cela implique dans le cas de chiffrement homomorphe dans la section I.4 et le chapitre II.

Sécurité des schémas basés sur les problèmes LWE et RLWE :
La difficulté des problèmes LWE et RLWE est très importante pour assurer la sécurité des schémas construits à partir d'eux. Il faut montrer que ces deux problèmes sont difficiles à résoudre. En effet, on utilise des réductions particulières, appelées réduction "pire-cas/moyen-cas". Et on montre que LWE (qui est donné pour des instances aléatoires, donc un cas "moyen"), est au moins aussi difficile à résoudre que toutes les instances d'une variante de SVP. La difficulté du problème LWE a été prouvée grâce à une réduction classique [Ajt96][BLP+13] (sous certaines conditions) et à une réduction quantique [Reg05][Reg09] (utilisant de l'algorithmique quantique en plus de l'algorithmique classique). Le problème RLWE a été prouvé difficile, sous certaines restrictions, par des réductions à une variante de SVP [LPR13].

* $f(x)$ a la particularité d'être un polynôme cyclotomique. Un polynôme cyclotomique associé à un entier naturel *n* est le polynôme unitaire dont les racines sont les racines primitives *n*-ièmes de l'unité. Son degré vaut $\varphi(n)$, où φ désigne la fonction indicatrice d'Euler. Il est à coefficients entiers et irréductible dans $\mathbb{Z}[X]$.
Pour des raisons de simplicité, $f(x)$ sera référencié comme étant un polynôme irréductible.

I.2. Chiffrement homomorphe avant Gentry

Dès 1978, Rivest, Adleman et Dertouzos [RAD78] avaient introduit la notion d'homomorphisme confidentiel (*privacy homomorphism*). Ils avaient conjecturé l'existence d'un algorithme où il serait possible d'évaluer une fonction publique sur les données chiffrées, mais sans rien révéler sur les entrées. Un tel schéma de chiffrement est resté un problème ouvert pendant une trentaine d'années. Pendant ce temps, de nombreux systèmes de chiffrement permettant d'effectuer soit des additions, soit des multiplications sur les chiffrés ont été proposés. On qualifie ces systèmes de schéma de chiffrement partiellement homomorphe (Partially homomorphic encryption – PHE).

L'exemple le plus simple est en fait le chiffrement RSA. Dans sa version déterministe, le produit de deux chiffrés est un chiffré valide du produit des deux messages clairs.

$$Enc_{RSA}(m_1) \,.\, Enc_{RSA}(m_2) = \; m_1^e \,.\, m_2^e \,.\, mod\; N = (m_1 . m_2)^e \; mod\; N = \; Enc_{RSA}\,(m_1 \,.\, m_2)$$

Cette version déterministe du RSA est qualifiée de non IND-CPA, ce qui est considéré comme une faiblesse dans beaucoup de contextes. D'autres versions probabilistes du RSA ont été proposées par la suite, mais ont perdu le caractère homomorphe de la version originale.

Je vais présenter dans la suite quelques schémas partiellement homomorphes. Le but de cette section n'est pas de dresser une liste exhaustive mais plutôt de donner un aperçu sur le domaine du chiffrement homomorphe avant la percée théorique de Gentry (propriétés homomorphes, sécurité, efficacité, etc.). Bien que partiellement homomorphes, ces schémas ont rendu possibles de nombreuses applications pratiques.

I.2.1. Système de Goldwasser-Micali

Goldwasser et Micali ont proposé en 1982 [GM82] le premier système de chiffrement sémantiquement sûr. Ce système est homomorphe par rapport à l'addition.

Génération des clés : $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$

Soit $N = pq$ avec p et q deux entiers premiers larges et x un non résidu quadratique modulo N tel que son symbole de Jacobi vaut +1.

$pk = (N, g)\; et\; sk = (p, q)$

Chiffrement : $c \leftarrow \text{Enc}(pk, m)$

Pour chiffrer $m \in \{0,1\}$, choisir aléatoirement un entier $r \in \mathbb{Z}_N^*$ et calculer

$c = x^m \, r^2 \, mod\; N$

Déchiffrement : $m \leftarrow \text{Dec}(sk, c)$

Pour obtenir m, vérifier si c est un résidu quadratique modulo N en utilisant la factorisation de N. Si c'est le cas, $m = 0$ autrement $m = 1$.

Propriétés homomorphiques :

Soient $m_1, m_2 \in \{0,1\}$, $c_1 = x^{m_1} \, r_1^2 \; mod\; N$ et $c_2 = x^{m_2} \, r_2^2 \; mod\; N$

$c_1 . c_2 = x^{\boldsymbol{m_1 + m_2}} \,.\, r_1^2 . r_2^2 \;\; mod\; N$

La somme $m_1 + m_2$ est obtenue en déchiffrant $c_1 . c_2$

Ce système possède une sécurité sémantique (i.e. IND-CPA) basée sur la difficulté du problème de la résiduosité quadratique qui consiste à décider si un élément aléatoire $y \in \mathbb{Z}_N^*$ est un carré ou non. Il est à noter qu'un message de l bits s'étend à un chiffré sur $l.|N|$ bits. Le facteur d'expansion (le rapport entre la taille des chiffrés et la taille des clairs) est égal à $|N|$ bits, ce qui est très grand. L'étape de chiffrement est simple, alors que le déchiffrement a une complexité $\tilde{O}(|N|^2)$. Ainsi, un compromis sécurité-efficacité apparaît ; en augmentant |N|, on a une meilleure sécurité au détriment de l'efficacité.

I.2.2. Système d'ELGamal

Le chiffrement ElGamal a été inventé par Taher ElGamal [ElG85]. Il représente l'alternative la plus courante au chiffrement RSA et possède un homomorphisme multiplicatif.

Génération des clés : $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$

Pour générer une paire de clés (pk, sk), l'utilisateur choisit un entier premier q et un générateur g du groupe cyclique $\mathbb{Z}_p^*$. Il génère un aléa $a \in \mathbb{Z}_q$ et calcule g^a.

$pk = (p, g, g^a)$ *et* $sk = a$

Chiffrement : $c \leftarrow \text{Enc}(pk, m)$

Pour chiffrer un message $m \in \mathbb{Z}_p^*$, l'utilisateur tire un aléa $b \in \mathbb{Z}_p$ et calcule $g^b \bmod p$ et $g^{ab} \bmod p$

$c = (c_{1,0}, c_{1,1}) = (g^b, m.g^{ab})$. Les valeurs sont modulo p.

Déchiffrement : $m \leftarrow \text{Dec}(sk, c)$

Pour déchiffrer $c = (c_{1,0}, c_{1,1})$, il suffit de calculer $m = c_{1,1}.(c_{1,0}^a)^{-1}$

Propriétés homomorphiques :

Soient c_1 et c_2 deux chiffrés respectifs de m_1 et m_2

$$c_1.c_2 = (g^{b_1}, m_1.g^{a.b_1}).(g^{b_2}, m_2.g^{a.b_2}) = (g^{b_1+b_2}, (\boldsymbol{m_1.m_2})g^{a(b_1+b_2)})$$

Le schéma ElGamal est prouvé IND-CPA sous l'hypothèse décisionnelle de Diffie-Hellman, dans le modèle standard. Toutefois, il faut remarquer que :

- Ce schéma est facilement adaptable à de nombreux groupes (pour lesquels le problème du logarithme discret est difficile) tout en préservant des propriétés homomorphiques, et en restant sûr (au sens des attaques à clairs choisis) sous l'hypothèse décisionnelle de Diffie-Hellman dans le groupe choisi.

  Une version homomorphe par rapport à l'addition a été présentée dans [CGS97] pour ne citer que cette variante.
- Avant l'étape de chiffrement, les messages doivent être encodés en éléments du groupe, ce qui requiert des conversions peu pratiques, limite la taille des messages et peut même supprimer les propriétés homomorphiques du schéma. Des améliorations ont proposé des schémas sans décodage.

Pour conclure, le schéma ElGamal est connu comme l'un des cryptosystèmes partiellement homomorphes les plus efficaces (facteur d'expansion de chiffrés = 2 et une complexité en $\tilde{O}(|q||p|)$ pour le chiffrement et le déchiffrement dans certains cas)

I.2.3. Système de Paillier

Paillier a proposé un cryptosystème [Pai99] qui reprend certaines idées d'Okamoto-Uchiyama [OU98]. Le système de Paillier est homomorphe par rapport à l'addition.
$\lambda(N)$ est la fonction lambda de Carmichael définie par $\lambda(N) = ppcm(p-1, q-1)$, $N = pq$ étant un module RSA standard.
La fonction $L_N(.)$ est définie sur l'ensemble $S_N = \{x < N^2 ; x = 1 \ mod \ N\}$ par $L_N(x) = (x-1)/N$

Génération des clés : $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$

Soit $N = pq$ avec p et q deux nombres premiers. Choisir un aléa $g \in \mathbb{Z}^*_{N^2}$ tel que $\text{pgcd}(L_N(g^{\lambda(N)}) \ mod \ N^2, N) = 1$

$pk = (N, g) \ et \ sk = \lambda(N)$ ou (p,q)

Chiffrement : $c \leftarrow \text{Enc}(pk, m)$

Pour chiffrer $m \in \mathbb{Z}_N$, choisir aléatoirement $r \in \mathbb{Z}^*_N$

$c = g^m r^N \ mod \ N^2$

Déchiffrement : $m \leftarrow \text{Dec}(sk, c)$

$m = \frac{L_N\left(c^{\lambda(N)} \ mod \ N^2\right)}{L_N\left(g^{\lambda(N)} \ mod \ N^2\right)} \ mod \ N$

Notons que $L_N\left(g^{\lambda(N)} \ mod \ N^2\right)$ est inversible du moment où on a choisi qu'il soit premier avec N.

Propriétés homomorphiques :

Soient c_1 et c_2 deux chiffrés respectifs de m_1 et m_2

$c_1.c_2 = g^{m_1+m_2}(r_1.r_2)^N \ mod \ N^2 = Enc(pk, m_1 + m_2 \ mod \ N)$

Ce schéma est prouvé sémantiquement sûr en se basant sur le problème de classe de résiduosité composite (la difficulté de distinguer les résidus d'ordre N modulo N² des non-résidus d'ordre N).
Le cryptosystème Paillier est le schéma partiellement homomorphe le plus connu ; une de ses applications phares est le vote électronique. Il figure aussi parmi les schémas les plus efficaces (facteur d'expansion = 2, Paillier a montré comment accélérer le déchiffrement en se basant sur le CRT, etc.)

I.2.4. Système de Boneh-Goh-Nissim

En 2005, Boneh Goh et Nissim [BGN05] ont proposé un système permettant d'évaluer un nombre arbitraire d'additions et une seule multiplication, présentant ainsi un premier pas vers les systèmes complètement homomorphes. Plus précisément, leur schéma permet d'évaluer un nombre arbitraire d'additions, une multiplication, puis un nombre arbitraire d'additions. Le nombre de multiplications est limité à 1 vu qu'elle est réalisée à l'aide d'un accouplement de Weil, qui change le groupe sous-jacent.
La sécurité de ce schéma est basée sur le problème décisionnel d'appartenance à un sous-groupe. Ce schéma souffre d'une forte expansion du message.
Dans [HF17], Herbert et al. ont présenté une version améliorée et plus efficace du [BGN05] offrant une profondeur multiplicative de 2 (au lien de 1 dans le schéma original).

I.2.5. Synthèse et Comparaison

Pour conclure cette partie sur le chiffrement PHE, je résume dans le tableau I-1 les caractéristiques des schémas présentés précédemment avec d'autres schémas partiellement homomorphes de l'état de l'art. Les résultats de ce tableau sont extraits des travaux de [FG07] et [Gue14]. Ils donnent des estimations et des ordres de grandeur afin d'avoir une comparaison globale des cryptosystèmes. Le lecteur intéressé est renvoyé au papier référence relatif à chaque schéma.

Tableau I.2. Etat de l'art des schémas partiellement homomorphes

Schéma	**Propriétés homomorphes**	**Expansion**	**Chiffrement** (Complexité)	**Déchiffrement** (Complexité)	**Sécurité** $\lambda = 128$	**Remarques**
GM	$\oplus$	$\|N\|$	$\|N\|$	$\|N\|^2$	$\|N\| = 2048$	Niveau bit
ElGamal	$\times$	2	$\|q\|\|p\|$	$\|q\|\|p\|$	$\|q\| = 320$ $\|p\| = 2048$	Encodage nécessaire
Benaloh	$+$	$\|N\|/\|l\|$	$\|N\|\|l\|$	$\geq \sqrt{N}$	$\|N\| = 2048$	$l \geq 2$ arbitraire
NS	$+$	$\|N\|/\|l\|$	$\|N\|\|l\|$	$\|N\|^3$	$\|N\| = 2048$	Amélioration de Benaloh
OU	$+$	3	$\|N\|^2$	$\|N\|^3$	$\|N\| = 2048$	Amélioration de Benaloh
Paillier	$+$	2	$\|N\|^2$	$\|N\|^2$	$\|N\| = 2048$	Amélioration de OU
DJ	$+$	$\frac{s+1}{s}$	$s^2 \|N\|^2$	$s \|N\|^2$	$\|N\| = 2048$	Généralisation de Paillier
Galbreith	$+$	3	$\geq DJ$		$\|N\| = 2048$ $\|E(\mathbb{Z}_{N^2})\| = 256$	DJ sur les courbes elliptiques

GM : Goldwasser-Micali [GM88] ElGamal [ElG85] NS : Naccache-Stern [NS98] Paillier [Pai99]
OU: Okamoto-Uchiyama [OU98] DJ: Damgärd- Jurik [DJ01] Galbreith [Gal02]

I.3. Chiffrement complètement homomorphe

En 2009, Craig Gentry [Gen09a][Gen09b] a présenté le premier schéma de chiffrement complètement homomorphe (FHE : Fully Homomorphic Encryption) permettant d'évaluer n'importe quelle fonction, i.e. des additions et des multiplications en nombre illimité, sur les données chiffrées.

I.3.1. Modèle de Gentry

Gentry a proposé la première construction d'un FHE basé sur les réseaux idéaux. L'idée de Gentry est de partir d'un système de chiffrement quelque peu homomorphe (SHE : Somewhat Homomoprhic Encryption) qui supporte un nombre limité d'opérations homomorphes

(particulièrement des multiplications). Son chiffrement utilise la notion d'erreur ou de bruit : un message venant d'être chiffré contient un certain niveau de bruit que la clé privée permet d'éliminer. Ce bruit croît au fur et à mesure des opérations et quand il dépasse un certain seuil, il empêche le déchiffrement. La seconde étape dite de "squashing" consiste à réduire la profondeur multiplicative du circuit de déchiffrement.
Une fois le circuit de déchiffrement simplifié, Gentry applique une procédure de rafraichissement (bootstrapping)*. Cette technique permet d'atténuer le niveau de bruit en le ramenant au niveau d'un message fraîchement chiffré. Le bootstrap évalue l'algorithme de déchiffrement dans le domaine du chiffré, retournant ainsi un nouveau chiffré correspondant au même message clair qu'avant l'opération.
Ces étapes forment un cadre de construction de schéma complètement homomorphe, illustré par la figure I.2. Ce cadre a servi de modèle pour les cryptosystèmes qui ont fait suite à celui de Gentry.

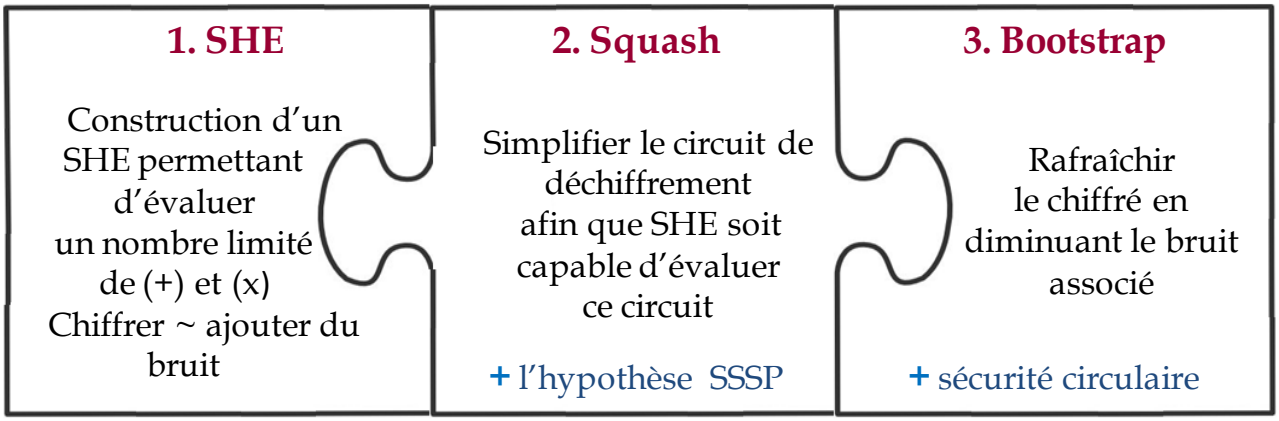

Figure I.2. Modèle simplifié de Gentry pour la construction d'un FHE

I.3.2. Description formelle et preuve de sécurité

Gentry a basé sa construction de schéma partiellement homomorphe sur les réseaux idéaux. Plus précisément, il a utilisé des idéaux d'anneaux de polynômes. La sécurité sémantique du schéma de Gentry est basée sur un problème dit "Ideal Coset Problem" qui peut être simplement vu comme le problème du décodage borné BDD.
Le chiffrement consiste à masquer le message (élément d'un anneau) par un bruit choisi dans un idéal I. Le message est alors chiffré en lui ajoutant ce bruit. La procédure de déchiffrement retire le bruit en effectuant une réduction modulo I.

* Le mot bootstrapping de bootstrap (tirants de bottes) fait référence aux aventures du baron de Münchhausen, qui se serait sorti d'un marécage en s'élevant dans les airs en tirant sur les tirants de ses bottes. Ça semble fou comme l'idée de Gentry d'ailleurs. Le mot est utilisé en informatique pour désigner le programme d'amorçage permettant de lancer le chargement d'un système d'exploitation (on trouve d'autres usages).
Dans le domaine de chiffrement homomorphe, sa première utilisation est due à Gentry. On utilisera les mots: rafraichissement, réamorçage, bootstrap et aussi bootstrapping pour désigner cette procédure.

Expliquons formellement ce que nous venons de décrire.
Soit R un anneau. Soit I un idéal de R, $I \subset R$ et B_I une base de I.

L'espace des messages m est $R \ mod \ B_I$.

Génération des clés : $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$

Soit J un idéal tel que I et J sont premiers entre eux ($I + J = R$).

On génère deux bases pour J : une "bonne" base B_J^{sk} et une "mauvaise" base B_J^{pk}

$pk = \left(B_I, B_J^{pk}\right) \ et \ sk = B_J^{sk}$

Chiffrement : $c \leftarrow \text{Enc}(pk, m)$

Pour chiffrer $m \in P \subset R \ mod \ B_I$, on choisit aléatoirement e dans l'idéal $m + I$

$c = e \ mod \ B_J^{pk}$

Déchiffrement : $m \leftarrow \text{Dec}(sk, c)$

Pour un chiffré c et avec la clé privée associée sk, on a : $m = \left(c \ mod \ B_J^{sk}\right) \ mod \ B_I$

Propriétés homomorphiques : $c \leftarrow Eval\ (pk, f, < c_1, c_2 >)$

Soient c_1, c_2 deux chiffrés associés à m_1, m_2

Addition homomorphe : $c_1 + c_2 = \ (m_1 + i_1 \ mod \ B_J^{pk}) + \ (m_2 + i_2 \ mod \ B_J^{pk})$

$$= (\ (\boldsymbol{m_1 + m_2}) + (i_1 + i_2\ ) \ mod \ B_J^{pk})$$

Multiplication homomorphe : $c_1 . c_2 = \ (m_1 + i_1 \ mod \ B_J^{pk}) . (m_2 + i_2 \ mod \ B_J^{pk})$

$$= (\ (\boldsymbol{m_1 . m_2}) + (m_1 . i_2 + m_2 . i_1 + i_1 . i_2\ ) \ mod \ B_J^{pk})$$

Les idéaux sont connus pour être stables par rapport à l'addition et à la multiplication, ce qui confère les propriétés homomorphes au schéma de Gentry. On remarque que le bruit augmente après chaque opération et qu'il est plus affecté par une multiplication qu'une addition. Après un certain nombre d'évaluations homomorphes, ce bruit va finir par dépasser un certain seuil et la procédure de déchiffrement retourne un résultat erroné. Donc, le schéma n'est pas encore totalement homomorphe. En d'autres termes, le schéma n'est pas encore "bootstrappable". Gentry a décrit une transformation de "squashing" permettant de simplifier le circuit de déchiffrement (ou le degré du polynôme de déchiffrement en fonction du modèle de calcul adopté). Le point crucial de la méthode de Gentry est d'obtenir un schéma qui peut à la fois évaluer des circuits d'une grande profondeur multiplicative et posséder un circuit de déchiffrement "assez simple" (ayant une profondeur multiplicative faible)*. Cette étape se fait toutefois au prix d'hypothèses supplémentaires, comme la somme de sous-ensembles peu denses SSSP (sparse subset sum problem).

Définition I.18. Schéma "bootstrappable" : Un schéma partiellement homomorphe est dit bootstrappable s'il est capable d'évaluer correctement :

- son propre circuit de déchiffrement ;
- et une porte multiplicative supplémentaire.

* En cas d'un modèle de calcul basé sur les polynômes : le point crucial de la méthode de Gentry est d'obtenir un schéma qui peut à la fois évaluer des polynômes de large degré et exprimer la procédure de déchiffrement comme un polynôme de degré "suffisamment faible".

L'étape de squashing a transformé le SHE en un SHE bootstrappable. Par conséquent, nous pouvons appliquer la procédure de rafraichissement qui consiste à nettoyer périodiquement le bruit pour éviter qu'il ne devienne trop important et nous obtenons un schéma totalement homomorphe. Le bootstrap nécessite une hypothèse de sécurité supplémentaire, connue sous le nom d'hypothèse de sécurité circulaire (circular security) ou plus généralement l'hypothèse au sens "key-dependant message" (KDM). La sécurité circulaire consiste à supposer qu'il est sûr de générer un chiffrement d'une clé privée sous sa propre clé publique. Il s'agit d'une hypothèse assez forte car elle ne découle pas de la notion classique d'indistinguabilité.

Théorème I.1.

SHE + profondeur (x) de Dec < d + Sécurité circulaire (ou KDM) ⇒ FHE

Afin de garantir la sécurité du schéma sous ces nouvelles hypothèses (sécurité circulaire, SSSP), il est souvent nécessaire d'augmenter la taille des paramètres qui le définissent.
Pour tout détail technique relatif aux notions précédemment décrites, je renvoie le lecteur aux articles [Gen09a] et [Gen09].

I.3.3. La procédure de rafraichissement

Comme nous l'avons précédemment décrit, les bruits inclus dans les chiffrés homomorphes voient leur taille augmenter avec les opérations effectuées. Ainsi, il est nécessaire d'appliquer régulièrement une procédure publique de rafraichissement sur les chiffrés.
Le bootstrapping est à ce jour le seul moyen connu pour obtenir un schéma totalement homomorphe.
Cette procédure consiste essentiellement à créer un nouveau chiffré c' de c de telle sorte que le bruit associé à c' soit réduit et c' déchiffre vers m. Toutes ces étapes s'effectuent dans le domaine homomorphe (à travers la couche de chiffrement). En d'autres termes, nous allons évaluer homomorphiquement l'algorithme de déchiffrement sur le chiffré de la clé secrète. Il est en effet possible de fournir à la procédure Eval un chiffré de la clé secrète c_{sk}, le circuit de déchiffrement du schéma, ainsi qu'un chiffré c (avec un gros bruit). Comme le montre l'équation **(A)**, le résultat de cette procédure est alors un nouveau chiffré c' du même message, mais qui aura été rafraichi par le circuit de déchiffrement.

$$Eval\big(Decryption\ circuit, c\ , Enc_{pk}(sk)\big) = c' \qquad \textbf{(A)}$$

L'évalué homomorphe d'un chiffré c est un autre chiffré c' qui déchiffre vers $Dec(sk, c) = m$.
Détaillons les étapes de cette procédure représentée :

- nous partons d'un SHE capable d'évaluer un nombre limité d'opérations et nous avons appliqué le squashing au circuit de déchiffrement ;
- pour un chiffré c, on considère $D_C(sk) = Decrypt_{sk}(c)$. Le $D_C(.)$ peut être exprimé comme un polynôme de faible degré (particulièrement en sk) ;
- on introduit dans la clé publique un indice supplémentaire *(an additional hint)* de la clé secrète $Enc_{pk}(sk)$. Autrement, on fournit un ensemble de vecteurs chiffrés de la clé secrète $sk = (sk_1, sk_2, \dots, sk_n)$ à la procédure d'évaluation. Ces chiffrés possèdent un niveau de

bruit très faible vu qu'ils viennent d'être chiffrés. L'hypothèse de la somme de sous-ensembles peu denses SSSP garantit cette étape ;

- enfin on évalue homomorphiquement le circuit de déchiffrement sur $D_C(.)$. Gentry a nommé cette étape "Recryption". Et nous obtenons un nouveau chiffré c' "frais". D'une manière générale, on considère le bootstrapping comme le concept et le recryption comme la procédure.

Bien que le réamorçage soit considéré comme l'idée ingénieuse de Gentry, il reste très coûteux et demande un calcul très complexe. Nous verrons dans la section I.4 les améliorations proposées à cette procédure.
Enfin, il faut noter que ce schéma de Gentry n'a plus aucun intérêt pratique aujourd'hui, dépassé en efficacité par les cryptosystèmes qui lui ont succédé. Cependant, le cadre qu'il a présenté est le seul permettant d'obtenir un schéma totalement homomorphe.

I.4. Chiffrement homomorphe : de la 1ère à la 3ème génération

Depuis l'introduction du premier cryptosystème complètement homomorphe par Gentry, plusieurs schémas ont vite fait leurs apparitions. La cryptographie homomorphe est devenue un domaine scientifique en pleine expansion. Chaque nouvelle proposition arrive avec une meilleure efficacité, en raffinant ou en changeant plusieurs concepts au niveau des preuves de sécurité et des structures mathématiques.
Nous présentons dans cette section l'évolution de la théorie du chiffrement homomorphe depuis 2009. Notre approche se concentrera sur la description des concepts que nous qualifierons comme les plus importants dans cette théorie et aussi pour la compréhension de la suite de ce manuscrit, plutôt que sur les détails mathématiques.

I.4.1. Une première classification

Nous avons vu jusqu'à maintenant trois types de schémas homomorphes : partiellement homomorphe, quelque peu homomorphe et totalement homomorphe. Rappelons que cette classification est basée sur les catégories de fonction f que le schéma est capable d'évaluer correctement. Nous rappelons les définitions de ces trois catégories et y ajoutons une nouvelle introduite par Brakerski, Gentry et Vaikuntanathan en 2012 [BGV12].

- Les schémas partiellement homomorphes ne peuvent évaluer qu'un nombre limité de fonctions. On parle d'homomorphisme additif pour les schémas pouvant évaluer uniquement des additions et d'homomorphisme multiplicatif pour ceux pouvant évaluer des multiplications sur les chiffrés.
- Les schémas quelque peu homomorphes (somewhat homomorphic encryption - SHE) peuvent évaluer sur les chiffrées des polynômes de petit degré, ce dernier étant fixé par les paramètres du schéma utilisé. Ils ont donc également des capacités calculatoires limitées, mais, contrairement aux schémas partiellement homomorphes, ils peuvent mélanger des additions et des multiplications et être transformés en des schémas totalement homomorphes.

- Les schémas homomorphes par niveaux (leveled homomorphic encryption - LHE) présentent moins de restrictions sur le nombre de multiplications possibles. En effet, pour tout entier L, il est possible de fixer les paramètres d'un tel schéma pour évaluer un polynôme de degré L. Leur principal inconvénient est que la taille des paramètres utilisés pour le schéma croît linéairement avec L.
- Les schémas totalement homomorphes (fully homomorphic encryption - FHE) sont des schémas pouvant évaluer des polynômes dont le degré est arbitraire sur les données chiffrées. Ils sont construits à partir des deux types de schémas précédents.

I.4.2. Les trois grandes familles de schémas homomorphes

Nous adoptons dans cette section la classification de [CCK+13] et divisons les schémas homomorphes en trois grandes familles en se basant sur les concepts mathématiques et les problèmes cryptographiques associés. La première famille incluant le schéma original de Gentry est basée sur les réseaux idéaux. En 2011, Brakerski et Vaikuntanathan constatent que la sécurité des cryptosystèmes homomorphes peut être basée sur des problèmes beaucoup plus simples, notamment LWE et RLWE. Depuis, le nombre de propositions par an croît d'une manière impressionnante, comme le montre la figure I.3. A partir de 2010, une nouvelle famille basée sur les entiers est introduite et s'est développée en parallèle. La figure 4 résume les schémas les plus importants de chaque famille. Comme nous pouvons le remarquer, la famille à base de (R)LWE est la plus riche. Nous découvrons dans la suite les raisons de cette richesse.

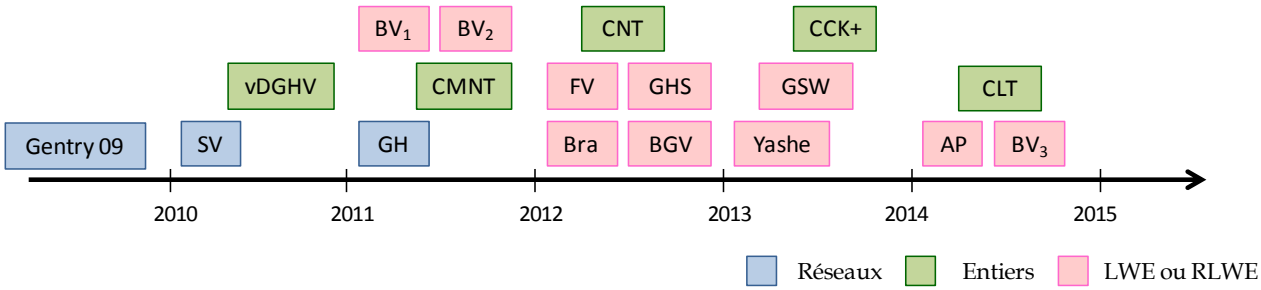

Figure I.3. Evolution de la cryptographie homomorphe

I.4.2.1. Schémas basés sur les réseaux

Les premiers schémas de chiffrement homomorphe proposés [SV10] [SS10] [GH11a] [GH11b] ont découlé de la construction de Gentry. Ils sont basés sur les réseaux idéaux et impliquent des notions mathématiques et cryptographiques complexes. La structure des réseaux idéaux garantit le caractère homomorphe et semble bien adaptée aux premiers candidats de FHE vu qu'elle supporte naturellement l'homomorphisme additif et multiplicatif. Les schémas basés sur les réseaux ont également des fonctions de déchiffrement "simples" (complexité logarithmique en n : dimension du réseau) ce qui les rend bien adaptés au bootstrapping. Aucun de ces schémas n'était LHE et ils utilisent tous le bootstrapping pour obtenir un cryptosystème totalement homomorphe.

Smart et Vercauteren [SV10] ont introduit une variante du schéma de Gentry avec des chiffrés et clés de tailles plus petites. C'est un schéma avec une conception plus simple des primitives de chiffrement et de déchiffrement, basée sur des réseaux idéaux spécifiques (principal ideal lattices). Cependant, une de ses limitations majeures est l'inefficacité de l'algorithme de génération des clés car il requiert des réseaux ayant des déterminants premiers. Cette primalité implique des ensembles de réseaux très restreints et devient incommode avec des paramètres de grande taille. Gentry et Halevi [GH11a] ont proposé un schéma sans squashing et ont éliminé ainsi l'hypothèse SSSP. D'autres optimisations ont été proposées dans [SS10] et une première implantation de schéma de Gentry amélioré a été présentée dans [GH11b]. Des durées très longues et des opérandes très grands ont été indiqués.
Les schémas basés sur les réseaux sont en nombre limité. Toutes leurs instanciations (prouvées sûres) utilisaient jusqu'alors des paramètres trop grands, des hypothèses de sécurité adhoc et une complexité théorique notable. Une utilisation pratique de ces schémas ne peut être alors envisageable.

I.4.2.2. Schémas basés sur les entiers

Une instanciation particulière du schéma de Gentry a été réalisée sur les entiers [vDGHV10], en se basant sur le problème du diviseur commun approché. Ce schéma suit le même modèle de base que celui de Gentry, mais sa construction est beaucoup plus simple du moment où il fonctionne avec des entiers et ne nécessite plus la complexité des réseaux idéaux. La sécurité de ce schéma se réduit au problème du diviseur commun approché [HG01].

Définition I.19: Problème du diviseur commun approché (Approximate-GCD problem)
Etant donné un ensemble $\{x_1, \dots, x_\tau\}$ où $x_i = pq_i + 2r_i$ pour $1 \leq i \leq \tau$, retrouver p.

Ce schéma se veut plus simple qu'efficace. L'idée est que la somme de deux nombres proches d'un multiple de p est également proche d'un multiple de p, de même pour le produit.
Nous présentons ci-dessous le schéma quelque peu homomorphe de DGHV basé sur un ensemble d'entiers publiques de la forme $x_i = p.q_i + r_i$ avec $0 \leq i \leq \tau$ et p est privé.

Pour un entier pair p, on utilise la distribution suivante sur des entiers de γ-bits

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

Génération des clés : $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
Générer un entier premier aléatoire de η-bits p.
Pour $0 \leq i \leq \tau$, échantillonner $x_i \leftarrow D_{\gamma,\rho}(p)$ selon la distribution $D_{\gamma,\rho}(p)$; Répéter jusqu'à x_0 est impair et $[x_0]_p$ est pair.
$pk = (x_0, x_1, \dots, x_\tau)$ et $sk = p$

Chiffrement : $c \leftarrow \text{Enc}(pk, m)$
Pour chiffrer $m \in \{0,1\}$, choisir aléatoirement un sous-ensemble $S \subseteq \{1,2,\dots,\tau\}$ et un entier $r \in (-2^\rho, 2^\rho)$
$c = \left[m + 2r + 2\sum_{i \in S} x_i\right] mod\ x_0$

Déchiffrement : $m \leftarrow \mathrm{Dec}(sk, c)$

Pour obtenir m, calculer $(c \ mod \ p) \ mod \ 2$.

$$\left[\left[m + 2r + 2\sum_{i\in S} x_i \mod x_0\right]_p\right]_2 = \left[\left[m + 2r + 2\sum_{i\in S} x_i - (p.q_0 + r_0).\lfloor c/x_0 \rceil\right]_p\right]_2$$

$$= \left[\left[m + 2r + 2\left(p\sum_{i\in S} q_i + \sum_{i\in S} r_i\right) - (p.q_0 + r_0).\lfloor c/x_0 \rceil\right]_p\right]_2$$

$$= \left[\left[m + 2r' + pq'\right]_p\right]_2$$

$$\underset{conditions}{=} m$$

Propriétés homomorphiques : $c \leftarrow Eval\ (pk, f, < c_1, c_2 >)$

Soient c_1, c_2 deux chiffrés associés à m_1, m_2

Addition homomorphe: $c_1 + c_2 = (\boldsymbol{m_1 + m_2}) + 2(r_1' + r_2') + p(q_1' + q_2')$

Multiplication homomorphe: $c_1.c_2 = (\boldsymbol{m_1.m_2}) + 2(r_1'm_2 + r_2'm_1 + 2r_1'r_2')$
$+p(q_1'm_2 + q_2'm_1 + pq_1'q_2' + 2r_2'q_1' + 2r_1'q_2')$
$= (\boldsymbol{m_1.m_2}) + 2r'' + pq''$

Afin d'obtenir un schéma complètement homomorphe, van Dijk et al. appliquent la procédure de bootstrapping, de façon analogue à celle de Gentry, c'est-à-dire en publiant un indice sur la clé secrète dans la clé publique et en évaluant homomorphiquement l'algorithme de déchiffrement sur le chiffré de la clé secrète.

L'avantage majeur de ce schéma réside en sa simplicité. Cependant, la taille des paramètres requis pour résister aux attaques connues sur le problème du diviseur commun approché AGCD est très grande; à savoir une clé publique de taille $\tilde{\mathcal{O}}(\lambda^{10})$ avec λ le paramètre de sécurité.

Plusieurs schémas basés sur les entiers ont été proposés afin d'améliorer l'efficacité du cryptosystème DGHV. Ces améliorations ont pu être réalisées en se basant sur différentes variantes de l'hypothèse de sécurité AGCD. Coran et al. [CNMT11] ont montré comment réduire la taille de la clé publique de $\tilde{\mathcal{O}}(\lambda^{10})$ à $\tilde{\mathcal{O}}(\lambda^{7})$ et ceci en conservant seulement une graine de la clé publique et puis en générant la clé totale au cours du traitement. Dans un autre travail, Coran et al. [CNT12] ont introduit une technique de compression des clés publiques, offrant une réduction de plusieurs ordres de grandeur de la taille de la clé publique (facteur de 80 par rapport à DGVH). Ils ont aussi adopté la technique de commutation de module initialement conçue pour les schémas basés sur LWE [BGV12]. Cette technique repose sur un nouveau problème décisionnel et consiste à convertir un chiffré modulo N en un chiffré modulo N′ plus petit. Le bruit est alors réduit automatiquement d'un facteur N/N′. Un choix judicieux de chaîne de modules permet d'obtenir un schéma par niveaux où l'augmentation du bruit est juste linéaire en la profondeur multiplicative du circuit. Cependant pour un circuit d'une profondeur multiplicative L, cette technique nécessite de travailler avec une clé publique environ L fois plus grande que dans le schéma classique. Un an après, Cheon et al. [CCK+13] ont importé la technique de batching/packing de [SV10, BGV12] qui utilise le théorème des

restes chinois CRT. Le batching permet à chaque chiffré de contenir plusieurs bits de données, i.e. de le transformer en un chiffrement par lots et d'effectuer des opérations en parallèle sur ces bits (sur le principe du SIMD: single instruction multiple data). Grâce à cette technique, le chiffrement peut opérer sur $\tilde{\mathcal{O}}(\lambda^2)$ messages à la fois, et l'expansion du chiffré passe de $\tilde{\mathcal{O}}(\lambda^5)$ à $\tilde{\mathcal{O}}(\lambda^3)$.
Dans [CLT14], Coran et al. ont appliqué la technique du module invariant introduite par Brakerski [Bra12]. Similaire au changement de modules, excepté que le même module est utilisé tout au long de l'évaluation homomorphe, elle permet d'obtenir un schéma homomorphe par niveaux dit à module invariant.
Les travaux visant à améliorer l'efficacité des schémas homomorphes à base d'entiers continuent à progresser [CS15]. Tous les nouveaux schémas proposés sont par niveaux et leur sécurité se réduit à une variante du problème AGCD. Notons que cette hypothèse de sécurité est encore étudiée et des attaques ont été présentées dans [CN12][CNT12].
L'intérêt principal de cette famille de cryptosystèmes homomorphes réside en sa simplicité. Côté efficacité, les premières instanciations s'exécutaient de manière équivalente et parfois moins bonne que les schémas à base de réseaux. Les versions revisitées et améliorées peuvent être considérées comme des candidats conséquents pour des schémas complètement/par niveaux homomorphes efficaces. Nous verrons ce point plus en détail dans le chapitre II. Une dernière caractéristique concernant cette famille et qui est particulièrement intéressante est que les améliorations apportées au schéma DGHV ont pu être réalisées en conservant le problème du grand diviseur approximé (et ses variantes). Alors que les révisions des autres schémas n'ont pu être faites qu'en changeant l'hypothèse de sécurité. Ceci peut s'expliquer par les réductions entre le problème LWE et le problème du AGCD exposées dans [CS15].

I.4.2.3. Schémas basés sur les problèmes LWE ou RLWE

Le premier schéma reposant sa sécurité sur le problème LWE a été présenté par Brakerski et Vaikuntanathan [BV11a]. Les auteurs ont introduit deux processus. Le premier est le processus de relinéarisation qui permet de baser le cryptosystème sur LWE plutôt que sur les hypothèses de complexité relatives aux idéaux. Le deuxième de réduction de module (ou de dimension) réduit naturellement la complexité de la fonction de déchiffrement et permet également de diminuer la taille des chiffrés. Il n'y a donc plus de nécessité de passer par l'étape de "squashing" de Gentry, qui lui avait coûté une hypothèse de sécurité supplémentaire. Brakerski et Vaikuntanathan ont aussi proposé une version basée sur le problème RLWE [BV11b].
Une lignée de schémas homomorphes par niveaux a été ensuite introduite [BGV12, Bra12, GHS12, FV12, BLLN13, GSW13, BV14, ASP14]. Chaque nouvelle construction apporte des techniques visant essentiellement à réduire la taille des paramètres et à augmenter la profondeur multiplicative.

Schéma de Brakerski, Gentry et Vaikuntanathan [BGV12] :
BGV est un cryptosystème asymétrique opérant au niveau bit. On trouve deux variantes de BGV : la première basée sur le problème LWE manipule des vecteurs d'entiers et la deuxième basé sur RLWE traite des polynômes à valeurs entières. On s'intéresse à la version polynomiale de BGV qui semble plus prometteuse en termes de performances.

On considère l'anneau de polynôme $\mathbb{R} = \frac{\mathbb{Z}[X]}{F[X]}$ avec $F(X)$ un polynôme irréductible de degré $d = 2^k$ et une succession de modules impairs $q_1 < \cdots < q_L$. Les sous-anneaux respectifs à ces modules $\mathbb{R}_{q_i} = \mathbb{R}/q_i\mathbb{R} = \{a_0 + a_1X + \cdots + a_{2k-1}X^{2k-1}\}$ présentent les anneaux de polynômes avec des coefficients entiers $\in]-q_i/2\,,\ q_i/2]$.

Génération des clés : $(pk, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$

La clé privée $sk \in \mathbb{R}$.

La clé publique $pk = a.sk + 2.e \in R_{qL}^N$ avec $N = \mathcal{O}(\log q_L)$, $a \in R_{qL}^N$ et e tiré aléatoirement selon une distribution Gaussienne discrète.

Chiffrement : $c \leftarrow \mathrm{Enc}(pk, m)$

Les entiers que nous manipulons doivent être chiffrés bit par bit.

Pour $m \in \{0,1\}$, le chiffré c est un couple de deux éléments de $\mathbb{R}_{q_L}$ dérivés de m, de la clé pk et d'un aléa.

Chaque chiffré c peut être transformé en un couple de deux éléments de $\mathbb{R}_{q_i}$.

Déchiffrement : $m \leftarrow \mathrm{Dec}(sk, c)$

La fonction de déchiffrement est un produit scalaire entre $c \in \mathbb{R}_{q_i}$ et la clé sk, suivi d'une réduction $mod\ q_i$ (dans $]-q_i/2\,,\ q_i/2]$) et d'une réduction $mod\ 2$.

Propriétés homomorphiques : $c \leftarrow Eval\ (pk, f, < c_1, c_2 >)$

Addition homomorphe: c'est une simple somme terme à terme modulo q_i. On appelle une procédure de mise à niveau en cas de besoin (c_1 et c_2 doivent appartenir au même niveau qi).

Multiplication homomorphe : elle correspond à un produit des deux chiffrés et nécessite un appel systématique à la procédure de mise à niveau. Le produit des deux chiffrés est un produit tensoriel $c_1 \otimes c_2$ et peut être décomposé en additions et multiplications modulaires de polynômes.

La procédure que j'appelle de mise à niveau ou de gestion de bruit (Noise management comme la nomme les auteurs de [BGV12]) permet de réduire le bruit quand il atteint un certain seuil mais aussi de remettre les chiffrés que l'on veut additionner ou multiplier au même niveau. Elle inclut une fonction de changement de clés ; amélioration de la relinéarisation de [BV11a] et une fonction de changement de module. Il faut noter que la majorité des idées présentes dans la construction du cryptosystème BGV sont issues de [BV11a]. La contribution majeure provient du raffinement du procédé de réduction de module, appelé dans [BGV12] changement de module et permettant de mieux gérer l'erreur contenu dans le chiffré. Sommairement, un chiffré c mod q peut être trasnformé en un chiffré valide c mod p, avec une opération simple de mise à l'échelle (p/q) suivie d'une opéartion d'arrondissment. Le changement de module utilse une une échelle décroissante de modules $(q_1, q_2, \ldots, q_L)$ et permet essentiellement de maintenir la quantité de bruit à un niveau constant.

La taille des chiffrés et le coût des opérations homomorphes (additions et multiplications) dépendent de la dimension de l'anneau $\mathbb{R}$ (i.e. le degré du polynôme $F(X)$) et de la taille des modules $\{q_i\}_i$. Un ordre de grandeur de ces paramètres est donnée comme suit : chaque bit de message clair est chiffré en un couple de polynômes de degré (dimension de $\mathbb{R}$) supérieur à 10^4 et de coefficients de taille (modules $\{q_i\}_i$) dépassant 200 bits. Afin de respecter un certain niveau

de sécurité et de garantir la gestion de bruit, la taille des paramètres croît avec le nombre de multiplications. Ainsi, le dimensionnement du cryptosystème BGV est essentiellement calibré par la profondeur multiplicative (le nombre de niveaux L).

Schéma de Brakerski [Bra12] :
A Crypto 2012, Brakerski a introduit un nouveau schéma qui diffère légèrement des cryptosystèmes homomorphes précédents dès lors qu'il propose de chiffrer les messages non plus dans le bit de poids faible, mais dans celui de poids fort, comme le faisait Regev dans [Reg05]. Brakerski a également présenté une nouvelle technique de gestion de bruit appelée technique d'invariance à l'échelle (Scale invariance).

Les avancées récentes :
Le schéma FV [FV12] est la version invariante de [BGV12] et YASHE [BLLN13] est la version invariante de [LTV12]*. Ces deux cryptosystèmes sont considérés parmi les schémas les plus efficaces et seront présentés à la section I.4.4.
A Crypto 2013, Gentry, Sahai et Waters introduisent un nouveau schéma [GSW13] qui repose également sur le problème LWE, mais qui a des propriétés uniques. En effet, la multiplication homomorphe est une simple multiplication de matrice et ne nécessite pas de changement de clé. De plus, nous verrons quand nous discutons des modèles de croissance du bruit (section I.4.5) que le bruit présent dans ce schéma se comporte différemment de celui des autres schémas.

I.4.3. Les techniques visant à améliorer l'efficacité

Si l'efficacité du chiffrement homomorphe ne cesse de s'améliorer, des points demeurent à exploiter. Les recherches dans le domaine de la cryptographie homomorphe suivent deux directions différentes mais complémentaires. La première concerne le bootstrapping et les schémas FHE et la deuxième s'intéresse aux schémas LHE.

I.4.3.1. Le réamorçage et les schémas complètement homomorphes

La procédure de réamorçage est certes très coûteuse. Que ce soit pour un bit ou n bits de données, cette procédure prend plusieurs minutes sur des processeurs actuels (à comparer aux millisecondes que prennent les autres opérations). Cependant, elle est nécessaire pour obtenir du chiffrement totalement homomorphe et elle est la seule connue aujourd'hui pour garantir cette caractéristique.
Plusieurs travaux visant à diminuer la durée de bootstrapping ont été proposés. Nous verrons les résultats au chapitre II quand nous abordons le chiffrement homomorphe en pratique. D'autres montrent comment minimiser le nombre d'appels au bootstrapping dans un circuit homomorphe [LP13][PV15]. Ainsi, en minimisant le nombre d'appels à cette procédure, ils donnent une manière efficace de réduire le temps de calcul homomorphe et d'améliorer l'efficacité des schémas concernés.

* Yashe et LTV sont plus précisément basés sur le problème NTRU [HPS98], ce qui diffère des problèmes LWE et RLWE. Etant donné que les schémas à base de RLWE et NTRU possèdent de nombreuses similitudes arithmétiques, et notamment Yashe et FV (les deux schémas auxquels nous nous référons le plus dans la suite du manuscrit), nous avons parlé de Yashe et LTV dans cette section consacrée aux schémas à base de (R)LWE.

Paindavoine et Vialla ont prouvé dans [PV15] que minimiser le nombre de réamorçages est un problème NP-complet et ont présenté un solveur qui obtient une bonne estimation du minimum de bootstrapping requis, ainsi que leur placement dans les calculs.

I.4.3.2. Les schémas homomorphes par niveaux

L'une comme l'autre des deux directions (améliorer l'efficacité du FHE ou du LHE) cherche à gérer le bruit contenu dans les chiffrés et diminuer sa croissance. Les travaux relatifs aux schémas homomorphes par niveaux ont présenté des nouvelles techniques ou raffiné d'autres déjà existantes. On cite le produit tensoriel, le changement de module, le changement des clés (ou relinéarisation), le batching (ou packing) et l'invariance à l'échelle. On va se contenter de présenter les deux dernières car les plus récentes et les plus efficaces. Il faut noter que toutes ces techniques ont été initialement conçues pour les schémas à base de (R)LWE et ensuite adaptées aux schémas à base d'entiers.

Batching :
En gros, le batching (ou packing) vise à améliorer l'efficacité à travers le parallélisme (SIMD). Dans [SV10, SV11], Smart et Vercauteren ont constaté que le théorème des restes chinois permet de chiffrer un vecteur de "messages clairs" (plaintext slots) au lieu d'un seul élément. Par suite, une seule opération homomorphe peut être évaluée à la fois sur les composantes du vecteur. Ils sont partis de cette observation pour proposer ce qu'on appelle "batch (or SIMD) homomorphic operations". En d'autres termes, cette technique propose d'utiliser un espace de textes clairs plus large en appliquant le théorème CRT. Ainsi, l'évaluation homomorphe d'une fonction f peut être exécutée sur k entrées différentes en parallèle, avec presque le même coût qu'évaluer f sur une seule entrée. Cette technique est utile dans le contexte de la cryptographie homomorphe du moment où elle exploite les bienfaits du parallélisme. Elle peut aussi être utilisée pour diminuer le surcoût lié aux calculs homomorphes, et ce en transportant efficacement les données entre les slots [GHS12].

L'invariance à l'échelle :
Comme nous l'avons déjà vu, les schémas par niveaux manipulent des chiffrés, qui sont des polynômes définis avec des coefficients modulo q. q est un entier de grande taille qui change à chaque fois que la procédure de changement de modules est appliquée. Cette technique donne un rendu très efficace, à savoir la réduction du niveau de bruit. Par contre, elle est très coûteuse car elle a besoin d'une nouvelle clé d'évaluation pour chaque niveau. En effet, afin d'atteindre une profondeur multiplicative de L-1, on a besoin de stocker et de manipuler L clés très grandes. Cette condition a été très contraignante lors de l'implantation des schémas faisant appel à cette technique.
A Crypto 2012, Brakerski a introduit une nouvelle technique permettant d'obtenir un schéma par niveaux pour les schémas homomorphes basés sur LWE [Bra12]. Il est parti de la technique de changement de module et a proposé de garder le même module tout au long de l'évaluation homomorphe. Brakerski a ainsi obtenu un schéma homomorphe à niveaux dit à module invariant. La technique de l'invariance à l'échelle est basée sur des chiffrés c (où sk est la clé secrète) tels que $\langle c, s\rangle = \lfloor N/e \rceil . m + e \ mod \ N$ avec e "petit", au lieu de $\langle c, s\rangle = m + 2.e \ mod \ N$

dans le schéma initial de Regev [Reg09] ; autrement dit le message est déplacé du bit de poids faible au bit de poids fort modulo N.

I.4.4. Aperçu sur les cryptosystèmes FV et Yashe

Commençons par définir les notions préliminaires nécessaires à la présentation des schémas FV et Yashe.

Soit d un entier positif et $R = \frac{\mathbb{Z}[X]}{\phi_d(x)}$ avec $\phi_d(x)$: un polynôme irréductible.

Les éléments de R sont les polynômes ayant des coefficients entiers et de degré inférieur à $n = \varphi(d)$ (φ: $indicatrice\ d'Euler$)

Pour un entier $q > 0$, on définit l'anneau quotient $R_q = \frac{R}{qR}$

Soit un entier $t > 0$, $r_t(q)$ est la réduction de $q\ mod\ t$ dans l'intervalle $[0, t)$

Soit $w > 1$ un entier ; $l_{w,q} = \lfloor log_w(q) \rfloor + 1$

Soit $\Delta = \left\lfloor \frac{q}{t} \right\rfloor$ alors $q = \Delta t + r_t(q)$

$[.]_q$ est la réduction modulo q dans l'intervalle $(-q/2\ , q/2]$ d'un entier ou d'un polynôme d'entiers (coefficients)

Un polynôme $a \in R_q$ peut être représenté dans la base w comme $\sum_{i=0}^{l_{w,q}-1} a_i w^i$ avec $a_i \in R$; les coefficients $\in (-w/2\ , w/2]$

On définit :

$$WordDecomp_{w,q}(a) = ([a_i]_w)_{i=0}^{l_{w,q}-1} \in R^{l_{w,q}}$$

$$PowersOf_{w,q}(a) = \left([aw_i]_q\right)_{i=0}^{l_{w,q}-1} \in R^{l_{w,q}}$$

$$< WordDecomp_{w,q}(a), PowersOf_{w,q}(b) > = ab\ (mod\ q)$$

χ_{key} et χ_{err} sont deux distributions discrètes, probabilistes et à support borné. En pratique, χ_{err} est une distribution gaussienne discrète et χ_{key} est une distribution dont les coefficients sont choisis dans l'ensemble $\{-1,0,1\}$.

B_{key} et B_{err} sont les bornes respectives de χ_{key} et χ_{err} .

$\|e\|_\infty < B_{err}$ pour $e \leftarrow \chi_{err}$ et $\|f\|_\infty < B_{key}$ pour $f \leftarrow \chi_{key}$

I.4.4.1. Le cryptosystème FV

Fan et Vercauteren reprennent le schéma BGV et transposent le problème sous-jacent de LWE à Ring-LWE en introduisant des nouvelles techniques et optimisations. La clé publique est une paire ($[-(as + e)]_q$, a) d'une instance RLWE (au signe près) et la clé secrète est un polynôme. Après une multiplication homomorphe, le chiffré est un vecteur formé de trois composantes au lieu de deux. Pour retrouver sa forme initiale, une opération supplémentaire de relinéarisation a été introduite (Figure I.4). La relinéarisation fait appel à une clé additionnelle notée evk. FV introduit deux paramètres additionnels, à savoir t et w. Si $t = 2$, les messages sont des bits. w est associé à la relinéarisation ; il détermine la complexité de cette opération et la taille de evk. En pratique, w est un entier de 32 bits ou de 64 bits.

Génération de paramètres : $params \leftarrow \text{KeyGen}(\lambda)$
Etant donnée λ, choisir un entier d qui définit R, deux modules q et t avec $1 < t < q$, deux distributions χ_{key} et χ_{err} et une base $w > 1$.
params = (d, q, t, χ_{key}, χ_{err}, w)
Génération des clés : $(pk, sk, evk) \leftarrow \text{KeyGen}(params)$
Echantillonner $s \leftarrow \chi_{key}$, $a \leftarrow R_q$ uniformément et $e \leftarrow \chi_{err}$.
Calculer $b = [-(as + e)]_q$
Echantillonner $\boldsymbol{a} \leftarrow R_q^{l_{w,q}}$ uniformément et $\boldsymbol{e} \leftarrow \chi_{err}^{l_{w,q}}$.
Calculer $\boldsymbol{\gamma} = \left(\left[PowersOf_{w,q}(s^2) - (e + a.s)\right]_q, a\right) \in R^{l_{w,q}}$
$(pk, sk, evk) = ((b, a), s, \boldsymbol{\gamma})$
Chiffrement : $c \leftarrow \text{Enc}(pk, m)$
L'espace des messages clairs est R/tR. On a $u \leftarrow \chi_{key}$ et $e_1, e_2 \leftarrow \chi_{err}$

$$\boldsymbol{c} = \left([\Delta[m]_t + bu + e_1]_q, [au + e_2]_q\right) \in R^2$$

Déchiffrement : $m \leftarrow \text{Dec}(sk, \boldsymbol{c})$
$\boldsymbol{c} = (c_0, c_1)$

$$m = \left[\left\lfloor \frac{t.[c_0 + c_1.sk]_q}{q} \right\rceil\right]_t \in R$$

Pour déchiffrer, calculer

$$\left[\left\lfloor \frac{t.[c_0 + c_1.sk]_q}{q} \right\rceil\right]_t = \left[\left\lfloor \frac{t.\left[[b.u + e_1 + \Delta.m]_q + [a.u + e_2]_q.sk\right]_q}{q} \right\rceil\right]_t$$

$$= \left[\left\lfloor \frac{t.[b.u + e_1 + \Delta.m + e_2.sk + a.u.sk]_q}{q} \right\rceil\right]_t$$

$$= \left[\left\lfloor \frac{t.\left[\left\lfloor \frac{t}{q} \right\rfloor.m + e.u + e_1 + e_2.sk\right]_q}{q} \right\rceil\right]_t$$

$$= [\lfloor\ m + \tilde{e}\ \rceil]_t$$

$$= m$$

Pour certaines valeurs de e et $\tilde{e} = \frac{e.u + e_1 + e_2.sk}{q}$, on trouve bien m tant que l'erreur $\tilde{e}$ ne dépasse pas un seuil.
Propriétés homomorphiques : $c \leftarrow Eval\ (pk, evk, f, < \boldsymbol{c_1}, \boldsymbol{c_2} >)$
Addition homomorphe :
Etant donné $\boldsymbol{c_1} = (c_{1,0}, c_{1,1})$ et $\boldsymbol{c_2} = (c_{2,0}, c_{2,1})$, on a :

$$\boldsymbol{c_{add}} = ([c_{1,0} + c_{2,0}]_q, [c_{1,1} + c_{2,1}]_q)$$

Multiplication homomorphe :

$$\tilde{\boldsymbol{c}}_{mult} = (c_0, c_1, c_2) \text{ avec } \begin{cases} c_0 = [\lfloor \frac{t}{q}.c_{1,0}.c_{2,0} \rceil]_q \\ c_1 = [\lfloor \frac{t}{q}.(c_{1,0}.c_{2,1} + c_{1,1}.c_{2,0}) \rceil]_q \\ c_2 = [\lfloor \frac{t}{q}.c_{1,1}.c_{2,1} \rceil]_q \end{cases}$$

$$\boldsymbol{c}_{mult} = \text{ReLin}\ (\tilde{\boldsymbol{c}}_{mult}, evk)$$

Relinéarisation : $\boldsymbol{S} \leftarrow$ ReLin $(\tilde{\boldsymbol{c}}_{mult}, evk)$
Soit $(\boldsymbol{b}, \boldsymbol{a}) = evk$ et $\tilde{\boldsymbol{c}}_{mult} = (\tilde{c}_0, \tilde{c}_1, \tilde{c}_2)$

$$\boldsymbol{S} = ([c_0 + < WordDecomp_{w,q}(c_2), \boldsymbol{b} >]_q, [c_1 + < WordDecomp_{w,q}(c_2), \boldsymbol{a} >]_q)$$

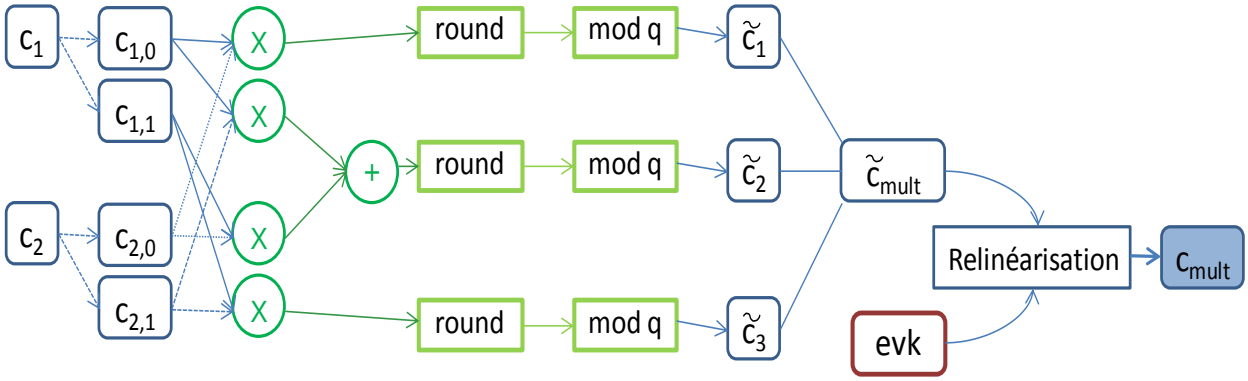

Figure I.4. Multiplication homomorphe du schéma FV

I.4.4.2. Le cryptosystème Yashe

Bos et al. [BLLN13] ont introduit un schéma homomorphe qu'ils ont nommé Yashe (Yet Another Somewhat Homomorphic Encryption) basé sur la version modifiée de NTRU de [SS11] et sur le schéma homomorphe présenté dans [LTV12]. Yashe fait aussi intervenir les deux paramètres t et w. Après une multiplication homomorphe, une opération de changement de clés est appliquée (Figure I.5).

Génération de paramètres : $params \leftarrow$ KeyGen(λ)
Etant donnée λ, choisir un entier d qui définit R, deux modules q et t avec $1 < t < q$, deux distributions χ_{key} et χ_{err} et une base $w > 1$.
params = (d, q, t, χ_{key}, χ_{err}, w)
Génération des clés : $(pk, sk, evk) \leftarrow$ KeyGen$(params)$
Echantillonner $f', g \leftarrow \chi_{key}$ et calculer $f = [tf' + 1]_q$. Si f n'est pas inversible *modulo* q, choisir un nouveau f'.
Calculer f^{-1} l'inverse de f *modulo* q et $h = [tgf^{-1}]_q$
$\boldsymbol{e}, \boldsymbol{s} \leftarrow \chi_{err}^{l_{w,q}}$.
Calculer $\boldsymbol{\gamma} = \left(\left[PowersOf_{w,q}(f) + \boldsymbol{e} + h.\boldsymbol{s} \right]_q \right) \in R^{l_{w,q}}$
$(pk, sk, evk) = (h, f, \boldsymbol{\gamma})$
Chiffrement : $c \leftarrow$ Enc(pk, m)
L'espace des messages clairs est R/tR.
Echantillonner $s, e \leftarrow \chi_{err}$
$c = [\Delta[m]_t + e + h.s]_q \in R$
Déchiffrement : $m \leftarrow$ Dec(sk, c)

$$m = \left[\left\lfloor \frac{t}{q} . [sk.c]_q \right\rceil \right]_t \in R$$

Propriétés homomorphiques : $c \leftarrow Eval\ (pk, evk, f, < c_1, c_2 >)$

Addition homomorphe :

$c_{add} = [c_1 + c_2]_q$

Multiplication homomorphe:

$$c_{mult} = \text{KeySwitch}\ (\tilde{c}_{mult}, evk)\ \text{ avec } \tilde{c}_{mult} = \left[\left\lfloor \frac{t}{q} . c_1 c_2 \right\rceil \right]_q$$

Changement de clé :

KeySwitch $S \leftarrow (\tilde{c}_{mult}, evk)$

$$S = [< WordDecomp_{w,q}(\tilde{c}_{mult}), evk >]_q$$

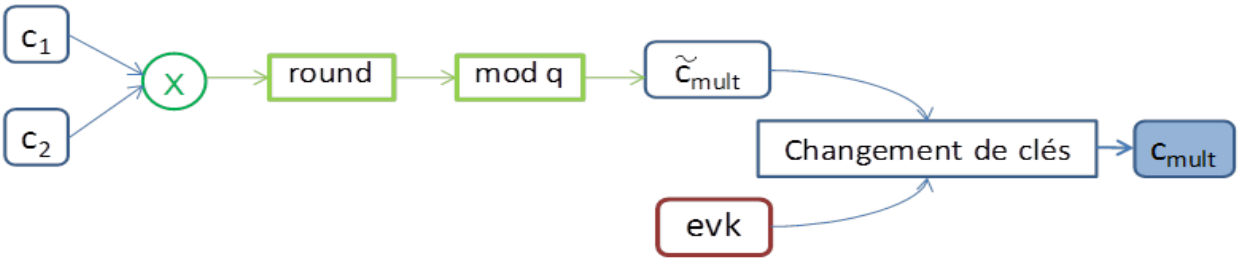

Figure I.5. Multiplication homomorphe du schéma Yashe

I.4.5. Choix et extraction des paramètres pour les schémas homomorphes à base du problème RLWE

Dans cette section, on s'intéresse aux schémas homomorphes à base du problème RLWE vu que cette variante "structurée" de LWE permet d'obtenir des primitives cryptographiques plus efficaces impliquant des durées de calcul plus courtes et des tailles des clés et des chiffrés plus petites.

I.4.5.1. Paramètres des schémas homomorphes à base du problème RLWE

Rappelons que le problème RLWE fait intervenir des anneaux de polynômes de la forme $R_q = \mathbb{Z}_q/f(x)$ avec $f(x)$ un polynôme irréductible. Il est défini autour de trois paramètres : la dimension n, le module q et le facteur d'erreur α (on parle aussi de $\sigma = \alpha.q$ l'écart type "standard déviation"). Dans le cas où il est appliqué au schéma de chiffrement, on rajoute la notion de niveau de sécurité quantifié par le paramètre de sécurité λ. Si de plus le schéma est homomorphe, on complète la liste avec la profondeur multiplicative L. Ces paramètres définissant les schémas homomorphes basés sur RLWE sont intimement liés et toute modification, même infime, est susceptible d'affecter de manière significative les performances du schéma et la difficulté des meilleures attaques. Assurer une efficacité optimale tout en maintenant un niveau de sécurité donné semble donc être un problème d'optimisation délicat.

La question de répondre à ce compromis efficacité-sécurité-fonctionnalité n'était pas posée et le but essentiel était de prouver la praticité de la cryptographie homomorphe.

Par contre, aujourd'hui cette question est devenue primordiale et constitue un axe à part dans le domaine de recherche de la cryptographie homomorphe. Des travaux sur la sélection des paramètres pratiques pour des schémas basés sur RLWE sont présentés dans [LN14, APS15, CS15, MBF16 et BF16]. Il s'avère que la dérivation des paramètres concrets définissant des primitives efficaces et sûres est un véritable défi qui nécessite des efforts supplémentaires.

I.4.5.2. Dérivation des paramètres

L'objectif est d'avoir un schéma sûr (attaques trop longues à réaliser) qui fonctionne correctement (après les opérations homomorphes, on veut déchiffrer le résultat correct) et efficacement (performances du schéma et profondeur multiplicative). Sans bruit ($\sigma = 0$), on a aucun souci pour que le déchiffrement fonctionne correctement mais aucune sécurité (tout est linéaire, les attaques sont triviales). Donc on voudrait augmenter σ pour atteindre un niveau de sécurité donné. Cette condition fait que le déchiffrement ne se passera pas correctement. D'où le compromis à trouver.

Sachant que :

- plus n est grand, plus les attaques seront difficiles mais plus le schéma sera lent ;
- plus $\frac{\sigma}{q} = \alpha$ est grand, plus la sécurité augmente mais plus les risques d'erreur au déchiffrement augmentent,
- σ doit vérifier $\sigma > 2\sqrt{n}$ pour avoir une sécurité prouvée [LPR13] mais en pratique il arrive qu'on ne s'impose pas un σ si grand puisque qu'on garantit que les attaques ne sont pas réalisables avec un grand n.

Dans [LN14], Lepoint et Naherig ont montré comment extraire ces paramètres pour les cryptosystèmes Yashe et FV. Leurs choix reposent sur une amélioration de l'approche de van de Pol et Smart et de la version complète de l'article de BKZ-2.0.

Nous avons vu dans la section I.1.3.4 que les problèmes basés sur les réseaux en général reposent principalement sur la difficulté de trouver un ou plusieurs vecteurs relativement courts dans un réseau à partir d'une base (apparemment) aléatoire/mauvaise, composée de vecteurs relativement longs et peu orthogonaux. Plus précisément, le problème RLWE a été prouvé difficile, sous certaines restrictions, par des réductions à une variante de SVP [LPR13]. Il existe des algorithmes pour résoudre les problèmes portant sur les réseaux :

- L'algorithme LLL de Lenstra, Lenstra et Lovasz : il résout une variante de SVP en temps polynomial et impose que deux coefficients diagonaux successifs ne décroissent jamais plus d'un facteur constant
- L'algorithme BKZ (Blockwise Korkine-Zolotar) : il est l'un des meilleurs algorithmes connus pour des problèmes sur les réseaux dans sa version 2.0. Il fonctionne à l'aide de deux types de routines. Dans un premier temps la base est (LLL)-réduite, puis un algorithme de réduction par bloc (de sous dimension < 100) est appelé. Chaque sous-bloc est résolu en faisant appel à un algorithme de crible, la solution est remontée dans l'algorithme par bloc, et ainsi de suite.

Pour évaluer les algorithmes de réduction des réseaux, on utilise le root Hermite factor. Lindner et Peikert [LP11] ont permis de faire l'analogie entre le root Hermite factor atteignable et le niveau de sécurité λ.

Dans un travail plus récent, Migliore et al. [MBF16] décrivent une procédure permettant de déterminer les paramètres définissant RLWE étant donné un niveau de sécurité λ et une profondeur multiplicative L. Sommairement, leur travail repose sur des tests de valeurs croissantes de n :

- calculer $\sigma > 2\sqrt{n}$ correspondant au minimum qui garantit une sécurité prouvée et donc celui qui entrainera le moins de problèmes au déchiffrement ;

- déterminer le plus grand q_{max} tolérable d'un point de vue sécurité étant donné n et λ ;
- déterminer le plus petit q_{min} tolérable pour que le schéma fonctionne correctement étant donnés n et L ;

Si $q_{max} > q_{min}$, on garde n. Sinon, il faut passer à un n plus grand.

Je résumerai au chapitre II un ensemble de paramètres pratiques pour FV et Yashe.
En conclusion, la dérivation de paramètres pour les schémas homomorphes à base du problème RLWE reste un sujet à explorer davantages et une automatisation de leur sélection va devenir nécessaire pour obtenir des primitives efficaces et sûres. En outre, cette sélection doit être prise en compte dans les implantations pratiques des cryptosystèmes homomorphes afin de réaliser des applications réelles.

I.4.6. Discussions

- Nous avons vu dans cette section les principaux cryptosystèmes homomorphes qui ont fait suite à la première construction de Gentry. Nous avons classé ces schémas en trois grandes familles. La famille des schémas à base des problèmes LWE et RLWE est bien évidemment la plus riche en termes du nombre de propositions. Elle suscite l'intérêt de la communauté scientifique vu qu'elle est beaucoup plus efficace (taille des clés, taille des chiffrées, performances, etc.) et qu'elle repose sur des hypothèses de sécurité standards et plus simples. Nous donnons dans le tableau I.2 un résumé des performances de quelques cryptosystèmes appartenant aux différentes familles. Les résultats présentés confirment notre conclusion sur l'efficacité des schémas à base de (R)LWE.

Tableau I.2. Quelques cryptosystèmes homomorphes

Schéma	Sécurité	Performances
Gentry	BDDP et SSSP	$\mathcal{O}(\lambda^{3,5})$ par porte pour le rafraichissement d'un chiffré
SV	PCP et SSSP	Génération des clés en $\mathcal{O}(logn\,.\,n^{2,5})$ avec n la dimension du réseau
Gentry amélioré	SVP et BDDP	Génération des clés en $\mathcal{O}(logn\,.\,n^{1,5})$ avec n la dimension du réseau
vDGHV	AGCD et SSSP	Taille de la clé publique $\mathcal{O}(\lambda^{10})$
Coran et al.	DAGCD et SSSP	Taille de la clé publique $\mathcal{O}(\lambda^5\log(\lambda))$
BV1	LWE	Taille de la clé d'évaluation $\mathcal{O}(\lambda^{2C}\log(\lambda))$ avec C un paramètre assurant le bootstrapping
BV2	RLWE	Génération des clés simplifiée
BGV	RLWE	Surcoût $\mathcal{O}(\lambda.\log(\lambda)\,.\,L^3)$

- Outre la classification des cryptosystèmes homomorphes en se basant sur la fonction qu'ils peuvent évaluer ou sur les concepts mathématiques et cryptographiques qui les définissent, nous pouvons les classer aussi en trois générations. Nous résumons dans le tableau 3 les principales caractéristiques de chaque génération. La plus ancienne est due à Gentry et n'est autre que la famille à base de réseaux. Cette génération n'est plus efficace et donc en cours de disparition. La deuxième et la troisième génération font partie de la famille des schémas à base des problèmes LWE et RLWE. En 2011, Brakerski et Vaikuntanathan ont proposé une nouvelle solution pour la construction des cryptosystèmes homomorphes. Les successeurs au schéma BV sont les plus efficaces des schémas des trois générations. La famille à base d'entiers est introduite et développée en parallèle à la deuxième génération. Les auteurs de [CS15] ont montré qu'elle est équivalente à la génération BV. Gentry, Sahai et Waters ont proposé en 2013 un nouveau principe différent des précédents introduisant ainsi la génération la plus récente. Cette dernière est moins efficace dans la plupart des cas. Mais quand on commence à évaluer des fonctions très compliquées, elle est meilleure que les autres, surtout au niveau du bootstrapping.

Tableau I.3. Les générations de la cryptographie homomorphe

<table>
<tr><th>1ère génération
Gentry'09</th><th>2ème génération
Brakerski et Vaikuntanathan'11</th><th>3ème génération
Gentry, Sahai et Waters'13</th></tr>
<tr><td rowspan="2">Peu efficace
Hypothèse de sécurité adhoc
Vulnérabilité : attaques sous-exponentielles [CDPR16]</td><td>Beaucoup plus efficace
Hypothèse de sécurité standard
Aucune faiblesse connue</td><td rowspan="2">Moins efficace en général
Hypothèse de sécurité légèrement meilleure</td></tr>
<tr><td>Schémas à base d'entiers</td></tr>
</table>

- Une autre approche est présentée dans la littérature des cryptosystèmes homomorphes. Cette approche définit un modèle de croissance du bruit selon les opérations et les schémas. Nous représentons le niveau de bruit l_i d'un chiffré c_i par un entier. Si un chiffré c_i est la sortie de l'algorithme de chiffrement et n'a pas encore subi d'évaluation d'opérations, son niveau est de 1. Ensuite, son niveau augmente au fur et à mesure des opérations exécutées lors de la procédure d'évaluation. Le bruit induit par les portes additives est logarithmique par rapport à celui généré par les portes multiplicatives. Dans la plupart des applications, il peut être considéré comme négligeable face à celui induit par les portes multiplicatives. Ainsi, soient c_1 et c_2 deux chiffrés de niveau de bruit l_1 et l_2 et $c_3 = Eval\ (c_1 + c_2)$. Le niveau l_3 de c_3 est alors $l_3 = \max(l_1, l_2)$. L'effet d'une porte multiplicative sur le niveau de bruit divise les schémas homomorphe en trois types :
  - Les schémas exponentiels, qui sont les schémas à modulo unique, parmi lesquels [Gen09b, vDGHV10, CMNT11, CCK+13, CLT14]. Dans ces schémas, $l_3 = l_1 + l_2$. Ainsi, l'évaluation d'un circuit de profondeur multiplicative D requiert que le niveau maximal que le schéma puisse gérer soit de 2^D. Cela devient très vite prohibitif, notamment du point de vue de la taille des paramètres, et en pratique le niveau maximal proposé par ces schémas est 2.

- Les schémas linéaires sont les schémas qui utilisent une suite de modulos, par exemple [BGV12, CNT12, LTV12]. Dans ces schémas, $l_3 = \max(l_1, l_2) + 1$.
- Les schémas asymétriques [GSW13]. Dans ce cas, $l_3 = l_1 + 1$.

Quand ce niveau de bruit atteint un seuil, qui dépend des paramètres du schéma considéré, le déchiffrement devient impossible. Il faut donc procéder avant ce seuil au bootstrapping.

- Les schémas de chiffrement homomorphe existants entrainent une expansion du chiffré tellement importante qu'elle rend inenvisageable l'envoi des données chiffrées avec un schéma homomorphe. Pour pallier cela, des solutions hybrides ont été proposées dans lesquelles les données sont transmises chiffrées sans expansion de chiffré (avec un schéma de chiffrement par blocs) puis déchiffrées de façon homomorphe avant d'être manipulées. Réaliser de telles évaluations homomorphes a fait l'objet de plusieurs publications [CCK+13][CLT14]. Nous présentons dans le chapitre II les résultats d'évaluations homomorphe de l'AES ou encore de chiffrement par blocs léger (Simon, Prince).

I.5. Applications

Il y a une quarantaine d'années, Rivest, Adleman et Dertouzos imaginaient un schéma où il serait possible de calculer sur les données chiffrées une fonction publique, mais sans rien révéler ni des entrées, ni de la sortie de la fonction évaluée, et les multiples applications de celui-ci.

Aujourd'hui, il devient possible de réaliser des calculs et tests statistiques ou des algorithmes d'apprentissage automatique simples en ne manipulant que des données chiffrées. Nous allons dans la suite identifier les applications pratiques de la cryptographie homomorphe en son état actuel et investiguer ce qu'il est possible de réaliser dans l'avenir.

I.5.1. Applications pratiques du chiffrement homomorphe

Les premiers prototypes utilisant du chiffrement homomorphe ont vu le jour récemment pour des applications sur des données médicales, biométriques ou de géo-localisation. Ils sont présentés par des groupes de Microsoft, IBM et Fujitsu.

Statistiques élémentaires :

La médecine du Big Data exploitant notre matériel génétique est un domaine en pleine croissance. En effet, la séquence unique de l'ADN d'un individu peut indiquer les prédispositions à des maladies, la tolérance à diverses substances, les traits potentiels de la descendance, le phénotype de l'individu, son origine, etc. Des séquences d'ADN de volontaires, stockées dans des biobanques permettent de percer les mystères de certaines maladies, prédire leur évolution et développer ainsi des traitements personnalisés. On estime aujourd'hui que plus de 1.600.000 personnes dans le monde avaient consenti à faire séquencer leur génome. Ainsi, la protection de ces données très sensibles est devenue une nécessité. Par exemple, la solution choisie par le CHUV (Centre hospitalier universitaire vaudois) de Lausanne, précurseur européen dans le domaine, est la cryptographie homomorphe.

Dans ce contexte, plusieurs travaux [LLN14][KL15][TJW+16][SCL+17] effectuant des calculs homomorphes sur des génomes ont été réalisés. Ils utilisent les cryptosystèmes Yashe, BGV et une bibliothèque dédiée au chiffrement homomorphe et implantant FV. Le travail [SCL+17] a

été présenté comme une solution au défi proposé par iDASH (integrating Data for Analysis, Anonymization, and Sharing) sur l'analyse sécurisée des génomes. En 2015 et 2016, le iDASH a accueilli deux concours internationaux visant à évaluer les limites du calcul sécurisé sur les génomes. Des participants du monde entier ont accepté le défi et ont réussi en 2015 à réaliser des évaluations homomorphes (calculs statistiques principalement) sur des séquences d'ADN réelles. En 2016, les auteurs de [SCL+17] ont analysé les génomes dans le but d'identifier l'existence d'une mutation. Des requêtes privées sur les génomes sont implantées et les résultats décrits dans [SCL+17] confirment que le chiffrement homomorphe peut être déployé en pratique.
Par ailleurs, Bos et al. [BLN14] ont présenté une analyse homomorphe des données de santé chiffrées visant à prédire une crise cardiaque.

Calculs numériques simples :
Les auteurs de [XCW+17] proposent une évaluation homomorphe de calculs arithmétiques sur les données chiffrées. Ils améliorent significativement les résultats de [CG15]. Ils ont implanté des opérations arithmétiques entières (additions, soustractions, multiplications et divisions) sur des données chiffrées par le schéma homomorphe BGV. Chaque opération est tout d'abord transformée en un circuit équivalent évaluable par BGV avant d'être exécutée sur les chiffrés.

Protection de la vie privée des consommateurs vis-à-vis de la publicité :
Actuellement, pour pouvoir cibler les individus, les publicitaires recueillent et croisent le maximum de données possibles émanant de notre activité, sur Internet notamment. Un service fondé sur la cryptographie homomorphe pourrait permettre de cibler les individus tout en s'assurant que les publicitaires ne connaissent rien de ces personnes. Ainsi, on arrive à construire une barrière de confidentialité entre les publicitaires et les consommateurs.

Exploration de données (Data Mining) :
Le data mining ou la fouille de données offre des techniques particulièrement efficaces permettant l'extraction d'informations significatives à partir de grandes bases de données. On est aujourd'hui à l'ère du Big Data où la volumétrie des données et leur variété suivent une croissance exponentielle. Ces données sont généralement personnelles et nécessitent d'être respectées et protégées. Dans ce contexte et dans un enjeu de protection de la vie privée, la cryptographie homomorphe est une solution envisageable. Il suffit d'implanter des variantes des algorithmes de fouille de données pour lesquelles les données sont chiffrées.

Intelligence artificielle :
Des travaux investiguant l'évaluation des modèles d'apprentissage automatique (machine learning) sur des données chiffrées ont été présentés dans [GLN12][DGL+16]. Les auteurs de [DGL+16] ont développé une intelligence artificielle appelée CryptoNets, un réseau neuronal qui peut traiter des données chiffrées sans la nécessité de les déchiffrer.

I.5.2. Chiffrement homomorphe comme bloc de base

Le chiffrement homomorphe peut être utilisé pour construire des outils cryptographiques tels que les preuves à divulgation nulle de connaissance (zero knowledge proof), les calculs multipartites sécurisés (MPC : Multiparty computation) et les signatures.
Prenons l'exemple du MPC qui groupe un ensemble de techniques cryptographiques permettant à plusieurs parties de calculer de façon interactive une fonction de leurs entrées respectives, sans dévoiler ces dernières, mais de façon à ne révéler que le résultat. Il a des applications innombrables : enchères en ligne, chiffrement décentralisé et collaboration entre des entités qui ne sont pas de confiance. Dans [LLS+16], les auteurs montrent comment utiliser du chiffrement homomorphe pour améliorer l'efficacité de leur protocole MPC. Le chiffrement homomorphe permet d'accélérer certaines étapes du MPC en réduisant les interactions entre les utilisateurs.

I.5.3. Applications futures

Une des applications futures de la cryptographie homomorphe est son adéquation aux besoins du cloud computing et de l'internet des objets.
Appliquer le chiffrement homomorphe dans un contexte du cloud computing permettra à la fois de répondre à un enjeu majeur de la société d'aujourd'hui qui est la protection de la vie privée et de permettre l'externalisation des calculs à des serveurs distants. En effet, il existe actuellement des techniques de chiffrement des machines virtuelles et des volumes de stockage mis en place par les prestataires des services du cloud, mais ces derniers ont toujours accès à des éléments en clair. Le chiffrement complètement homomorphe permet d'éliminer ce problème en effectuant des opérations directement sur des données cryptées.
Ainsi, l'utilisateur peut envoyer une requête chiffrée au serveur cloud pour effectuer une opération simple ou complexe sur les données chiffrées et obtenir un résultat chiffré qui sera déchiffré avec sa propre clé. Le champ d'application peut encore s'élargir touchant les domaines de la santé, des applications mobiles, des réseaux sociaux, des objets connectés et des villes intelligentes.

En conclusion, la cryptographie complètement homomorphe est encore dans une phase de recherche d'applications pratiques et une amélioration de l'efficacité est encore nécessaire sur tous les plans théoriques et pratiques. Côté applicatif, il serait possible de trouver des applications qui n'exploiteraient pas les faiblesses mais les forces du chiffrement homomorphe, ou à l'inverse de concevoir de nouvelles applications pratiques qui sont adaptées aux contraintes actuelles des schémas FHE.

I.6. Conclusion

L'état de l'art en cryptographie homomorphe a significativement changé ces cinq dernières années depuis l'introduction du premier système complètement homomorphe. En 2009, nous disposions d'un schéma uniquement théorique, dont la sécurité reposait sur des hypothèses fortes et son efficacité est très peu satisfaisante. A l'heure actuelle, nous parlons de trois générations de cryptosystèmes homomorphes. La première génération est déjà en cours de disparation et nous en gardons le modèle de base de Gentry, le seul connu aujourd'hui pour la

construction d'un schéma complètement homomorphe. Cependant, les schémas les plus récents basés sur des problèmes plus classiques sont au centre de la communauté scientifique ; avec un intérêt particulier aux cryptosystèmes de la seconde génération. La plupart de ces schémas possèdent des implantations prometteuses. En particulier, des évaluations homomorphes de différents circuits (AES, Simon, Prince) s'exécutent en quelques minutes sur des ordinateurs actuels.

Nous allons présenter au chapitre II les différentes implantations des cryptosystèmes homomorphes et des primitives utilisées pour la construction de tels schémas. Nous nous intéresserons aux implantations logicielles mais aussi aux implantations sur des plateformes alternatives (GPU, FPGA, ASIC). Une comparaison d'efficacité des travaux de l'état de l'art sera faite.

Chapitre II. Etat de l'art des implantations dédiées aux cryptosystèmes homomorphes

Dans ce chapitre, nous nous intéressons à l'aspect pratique du chiffrement homomorphe. Plusieurs schémas homomorphes ont été implantés, avec une efficacité croissante [CMNT11, PBS11a, CNT12, GHS12, FSF+13, BLLN13, LN14, HS14, HS15, LP16]LN14LN14LN14LN14. Néanmoins, parmi ces implantations, peu sont publiques [PBS11a, LN14, HS14, HS15, LP16].
Je vais présenter tout d'abord une vue d'ensemble des algorithmes de chiffrement homomorphe implantés sous forme logicielle, exécuté sur un processeur généraliste. Cette vue d'ensemble me permettra de comparer les performances et de donner un ordre de grandeur de leur potentiel pratique. Ensuite, je me focaliserai sur les implantations des algorithmes homomorphes sur des plateformes alternatives (GPU, FPGA, ASIC). Enfin, je conclurai ce chapitre avec un bilan et une synthèse sur les travaux existants.

II.1. Implantations logicielles pour les schémas à base de réseaux et d'entiers

II.1.1. Schémas à base de réseaux

Smart et Vercauteren ont présenté la première implantation [SV10] du système de Gentry basée sur les réseaux idéaux principaux impliquant un réseau avec un déterminant primaire.
Ils n'ont pas pu implanter la procédure de bootstrapping et leur construction correspond à un système quelque peu homomorphe. En effet, la condition de primalité du déterminant implique une grande complexité au niveau de la procédure de génération des clés et rend impossible la génération de clés pour des dimensions supérieures à $2048 = 2^{11}$. Or pour pouvoir réaliser le bootstrapping la dimension minimale du réseau doit être 2^{27}. Pour des réseaux de petites dimensions, la génération de clés prend plusieurs heures (calculer plusieurs résultantes et vérifier la primalité).
En 2011, Gentry et Halevi [GH11] ont présenté la première implantation du schéma complètement homomorphe de Gentry. Ils ont suivi la même direction que Smart et Vercauteren en ajoutant des améliorations et optimisations. Ils ont testé leur implantation avec des réseaux de plusieurs dimensions correspondant à différents niveaux de sécurité. Le tableau II.1 résume les performances pour la plus petite configuration "toy" et la plus grande qu'ils ont testé. Pour la grande configuration, leur implantation exige des clés publiques de taille 2,3 GB. La génération de clés dure 2,2 heures. L'opération de réamorçage, nécessaire après chaque multiplication, prend 30 minutes et nécessite une grande quantité de mémoire. Le paramètre de sécurité est fixé à 72. Bien que cette implantation nécessite un coût prohibitif en termes de durées et de ressources (taille des clés, des chiffrés, etc.), elle a le mérite d'être la première réalisation pratique d'un schéma FHE.

Tableau II.1. Performances de l'implantation de Gentry et Halevi d'un schéma à base de réseaux

Paramètres*	$\|pk\|$	Génération de clés	Chiffrement	Déchiffrement	Opération de "Recrypt"
n=512	17 MB	2,5 s	0,19 s	< 0,01 s	6 s
n= 32768	2,3 GB	2,2 h	3 min	0,66 s	31 min

* n: dimension du réseau

Perl et al. [PBS11a] ont implanté le schéma de Smart et Vercauteren en y ajoutant la procédure de bootstrapping. Cette implantation nommée HCRYPT est en logiciel libre [PBS11b].

II.1.2. Schémas à base d'entiers

La plupart des conceptions de schémas à base d'entiers était accompagnée d'implantations logicielles. Ces dernières regroupent des implantations "preuve de concept" destinées à valider les résultats théoriques et d'autres plus efficaces pouvant être déployées en pratique. Deux implantations du schéma vDGHV sont présentées dans [CT12] et [CNT12]. La première est libre et développée avec Sage; la deuxième propose une évaluation homomorphe de l'AES. Je résume dans le tableau II.2 les performances d'un schéma plus récent, celui à module invariant de Coran et al [CLT14]. Les auteurs ont également évalué l'AES de façon homomorphe. Pour 80 bits de sécurité, une évaluation complète du circuit sur 1875 blocs de 128 bits nécessite 102 heures, ce qui donne un temps relatif de l'ordre de 3 minutes de calcul par bloc de données.

Tableau II.2. Performances du schéma à base d'entiers de Coran et al.

Sécurité	$\|pk\|$	Génération de clés	Chiffrement	Déchiffrement	Opération de "Recrypt"
$\lambda = 42$	3,2 MB	0,5 s	< 0,1 s	< 0,1 s	0,1 s
$\lambda = 62$	704 MB	5 min	3 s	0,2 s	2,8 s
$\lambda = 80$	100 GB	213 h	5 min	24 s	4,6 min

II.2. Implantations logicielles pour les schémas à base des problèmes LWE et RLWE

Nous avons montré au premier chapitre que les schémas à base des problèmes LWE et RLWE sont les plus efficaces d'un point de vue théorique, formant ainsi la famille la plus étudiée par la communauté cryptographique. Les implantations de ces schémas sont aussi nombreuses. Certaines utilisent des bibliothèques logicielles connues (GMP, FLINT, NTL, …) et d'autres non.

II.2.1. Premières implantations

Dans [NLV11], Lauter et al. ont proposé une implantation du schéma [BV11b] en utilisant le logiciel de calcul formel Magma. Leur implantation correspond à un schéma quelque peu homomorphe et est juste présentée comme une preuve de concept visant à étudier la praticité du chiffrement homomorphe. Elle a été réalisée pour plusieurs paramètres et a été appliquée pour faire des calculs statistiques simples de façon homomorphe.

La première évaluation homomorphe d'un circuit non trivial (AES) est présentée dans [GHS12]. Gentry, Halevi et Smart ont implanté une variante du schéma BGV [BGV12] sans bootstrapping en se basant sur la bibliothèque NTL. Ils ont introduit plusieurs optimisations afin de pouvoir évaluer AES-128 dans le domaine chiffré. Ils ont dû utiliser un serveur avec 256 Go de mémoire vive pour leur évaluation homomorphe qui a duré 36 heures. Ces chiffres sont énormes et rendent ce travail inutilisable en pratique. En 2015, les mêmes auteurs [GHS15] ont proposé une mise à jour de cette implantation. Grâce à des améliorations à plusieurs niveaux détaillées dans [GHS15], une évaluation homomorphe de l'AES-128 est réalisée en 3 heures sur une machine avec 3 GB de de mémoire vive.
Cette efficacité croissante est aussi notée dans d'autres implantations [HS14, FSF+13] des variantes du cryptosystème BGV [BGV12]. L'implantation la plus connue du schéma BGV est la bibliothèque HElib [HS14]. Elle est présentée dans la section II.2.2.1.
J'ai aussi réalisé une implantation du schéma BGV en Sage. Ce travail m'a permis de bien comprendre l'enchaînement des calculs dans la construction d'un tel schéma et d'évaluer les différentes primitives qui le définissent. Les résultats de mon prototype logiciel sont décrits au chapitre III. Sur la base de ces résultats, je pourrai expliquer les choix que j'ai fait pour concevoir et implanter des accélérateurs matériels dédiés au chiffrement homomorphe et comparer les performances des implantations matérielles et logicielles.

II.2.2. Bibliothèques logicielles permettant de faire du chiffrement homomorphe

II.2.2.1. Bibliothèque HElib

La bibliothèque HElib de Halevi et Shoup met en œuvre le schéma BGV avec des améliorations apportées depuis la publication du schéma telles que le batching et le bootstrapping. Elle est développée en C++ et utilise les bibliothèques GMP et NTL. La documentation de HElib (disponible sur github [HS14]) donne les premières bases pour l'utiliser et résume certaines de ses performances. La bibliothèque est divisée en deux couches (figure II.1) : une couche réservée au calcul mathématique (math layer) et une couche pour les primitives cryptographiques (crypto layer). Les classes bluestein et CModulus évaluent les polynômes dans le domaine FFT en utilisant NTL, alors que les classes PAlgebra et PAlgebraMod sont consacrées aux opérations de codage et décodage. La classe Double-CRT, la plus importante de la couche mathématique, manipule les polynômes dans leurs représentations Double-CRT. La couche cryptographique implante le schéma BGV.
L'opération de réamorçage est introduite dans HElib en 2015 [HS15]. Son implantation la plus récente dure environ 6 minutes sur un Intel Core i5-3320M (2,6 GHz). En plus, une évaluation homomorphe de l'AES en utilisant HElib demande une durée de 3 heures au lieu des 36 heures de [GHS15].
HElib est devenue une référence pour évaluer le chiffrement homomorphe car elle implante un schéma totalement homomorphe intégral et elle inclut des techniques efficaces de batching et d'autres optimisations. Par exemple, des groupes d'IBM, Microsoft et Stanford/MIT ont eu recours à HElib pour des applications relatives à l'analyse des génomes [KL15]. HElib est aussi utilisée pour réaliser des opérations arithmétiques simples sur des données chiffrées [XCW+17].

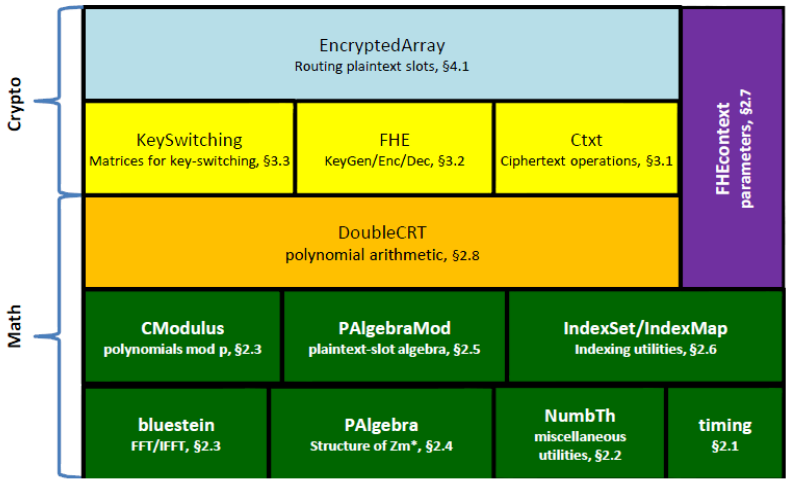

Figure II.1. Digramme de base de HElib [Wen15]

II.2.2.2. NFLlib

NFLlib [ABG+16a][ABG+16b] est une bibliothèque open source développée en C++ et dédiée à la cryptographie à base de réseaux euclidiens. Elle peut ainsi être utilisée pour implanter plusieurs schémas homomorphes. NFLlib inclut des optimisations algorithmiques (théorème des restes chinois CRT, transformation NTT optimisée) et des optimisations au niveau du développement. Cette bibliothèque a la particularité d'être conçue pour les anneaux de polynômes de forme spécifique $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ ce qui la rend plus efficace que les bibliothèques génériques comme NTL et FLINT. Le tableau II.3 compare les performances de deux implantations du schéma FV sur un processeur Intel Core i7-4650U Processor, 1,7 GHz : la première utilise FLINT et la deuxième NFLlib (dans ce cas les auteurs remplacent FLINT dans le travail de [LN14] par NFLlib). Dans cette direction, une proposition intéressante évoquée dans [ABG+16a] est de remplacer NTL dans HElib par NFLlib.

Tableau II.3. Performances de deux implantations différentes du schéma FV pour les paramètres (n=4096, q=124 bits)

	Chiffrement	**Déchiffrement**	**Add. Hom.**	**Mult. Hom.**
FV avec FLINT	26,7 ms	13,6 ms	1,1 ms	91,2 ms
FV avec NFLlib	0,9 ms	0,9 ms	0,01 ms	17,2 ms

II.2.2.3. Seal

Seal (Simple Encrypted Arithmetic Library) [CLP+17a] [CLP+17b] est une bibliothèque open source développée en C++ par Microsoft Research. Sa première version date de 2015, la version v2.2 étant apparue en avril 2017. Seal s'adresse à la fois aux experts et non experts du domaine et est fournie avec une documentation claire et des exemples pratiques [CLP+17a]. Elle est basée sur une version modifiée du schéma FV. Le tableau II.4 donne les durées de l'opération de multiplication homomorphe pour plusieurs valeurs de n et q sur une machine fonctionnant à

2 GHz. Seal est utilisée dans plusieurs applications, réalisant notamment des analyses et calculs statistiques sur les génomes [TJW+16] [SCL+17].

Tableau II.4. Durée de la multiplication homomorphe implantée dans Seal pour plusieurs valeurs de (n,q)

n	q	Mult. Hom.
1024	35	3,46 ms
2048	60	8,51 ms
4096	116	24,89 ms
8192	226	107,1 ms

II.2.2.4. FHEW

La bibliothèque FHEW a été introduite par Ducas et Micciancio en 2015 [DM15]. Elle est développée en C et C++. Elle utilise la bibliothèque FFTW pour calculer la FFT plus efficacement. L'opération de bootstrapping implantée sous FHEW dure moins d'une seconde sur un processeur Intel Core 64 bits fonctionnant à 3 GHz.

II.2.2.5. $\Lambda \circ \lambda$

La bibliothèque $\Lambda \circ \lambda$ [CP16a][CP16b] est développée majoritairement en Haskell avec du C++. Elle est open source et comporte 4991 lignes de code. Elle est construite autour de 4 niveaux : niveau pour les entiers (integer layer), niveau tensoriel (tensor layer), niveau cyclotomique (cyclotomic layer) et niveau cryptographique (cryptography layer). Les deux premiers niveaux implantent les fonctions nécessaires pour accélérer les opérations de chiffrement et de déchiffrement. Le niveau cyclotomique est consacré à l'échantillonnage Gaussien et à quelques opérations relatives à la cryptographie à base de réseaux euclidiens. Finalement, le niveau cryptographique implante le chiffrement homomorphe.

II.2.2.6. Remarques

- Nous venons de présenter cinq bibliothèques conçues pour faire du chiffrement homomorphe (dont certaines visent la cryptographie à base de réseaux euclidiens) autres que les bibliothèques génériques (FLINT, NTL, …). Comparer les performances de ces implantations afin de déterminer la plus efficace ne serait pas correct. En effet, les schémas implantés ne sont pas les mêmes, les paramètres définissant chaque schéma sont très différents et le choix des paramètres est basé sur des hypothèses et des algorithmes de réduction propres à chaque bibliothèque. Cependant, toutes ces bibliothèques sont open source et les utilisateurs ont accès aux codes sources et aux articles de référence ; ils peuvent choisir la bibliothèque la plus appropriée à leurs applications. J'ai résumé dans le tableau II.5 les caractéristiques générales de ces bibliothèques.

- Concernant les durées du bootstrapping, si Ducas et Micciancio ont proposé une implantation permettant un calcul en moins d'une seconde, leur procédure est applicable uniquement au schéma [GSW13]. De plus, elle ne concerne que les chiffrés d'un message d'un seul bit [DM15]. La procédure implantée dans HElib prend, elle, environ 6 minutes, mais pour un espace de clairs beaucoup plus grand et des chiffrés regroupant des vecteurs de messages [HS15]. Le coût amorti par bit d'un réamorçage exécuté sous HElib est à peu près du même ordre de grandeur que celui de [DM15]. Cette technique a été améliorée par [CGGI16], arrivant à un réamorçage en 0,1 seconde, mais toujours avec les mêmes restrictions.

Tableau II.5. Caractéristiques générales des différentes bibliothèques de chiffrement homomorphe

Bibliothèque	**HElib**	**FHEW**	**$\Lambda \circ \lambda$**	**Seal**	**NFLlib**
Année	2013	2014	2015	2015	2016
Bibliothèques associées	NTL, GMP	FFTW3	Aucune	Aucune	Aucune
Langage	C et C++	C et C++	Haskell et C++	C++ et C#	C++
Plateforme	Linux	Linux	Linux et Windows	Linux et Windows	Linux
Problèmes associés	RLWE	LWE, RLWE	RLWE	RLWE	RLWE

II.2.3. Implantations plus récentes

Dans cette section, nous nous intéressons à deux schémas homomorphes récents : FV [FV12] et Yashe [BLLN13]. Ce sont deux cryptosystèmes à base de réseaux et à module invariant. Comme nous l'avons déjà vu, il y a des bibliothèques qui ont implanté le schéma FV. Dans leur article introductif [BLLN13], Bos et al. ont également réalisé une implantation de Yashe et ont décrit les résultats pratiques.

Lepoint et Naherig [LN14] ont effectué une implantation en C++ des deux schémas en utilisant FLINT. Cette implantation disponible sous github [Lep14] a permis de faire une comparaison pratique de FV et Yashe et d'évaluer leurs forces et faiblesses respectives. Le tableau II.6 résume les performances de chaque schéma. Ces résultats montrent que Yashe est meilleur que FV sauf pour la génération des clés et l'addition homomorphe et que les différences sont essentiellement dues à la structure des chiffrés (chiffrés de FV à deux éléments de Rq contre chiffrés de Yashe à un seul élément). Le tableau II.7 donne des exemples de tailles des chiffrés et des clés pour le schéma FV. Lepoint et Naherig ont aussi évalué de façon homomorphe le schéma de chiffrement par blocs léger Simon.

Tableau II.6. Performances de FV et Yashe (Intel Core i7-2600, 3,4 GHz)

Schéma	**λ**	**KeyGen**	**Enc**	**Dec**	**Add. Hom.**	**Mult. Hom.**
FV	80	0,2 s	34 ms	16 ms	1,4 ms	148 ms
Yashe	80	3,4 s	16 ms	15 ms	18 ms	49 ms

Tableau II.7. Taille des clés et des chiffrés pour le schéma FV

n	q	sk	pk	chiffré
4096	127	63,5 KB	315,5 KB	127 KB
8192	300	300 KB	3 MB	600 KB
16384	700	1,4 MB	31,3 MB	2,8 MB

II.2.4. Evaluation homomorphe des circuits connus

Réaliser des évaluations homomorphes de circuits connus ou le chiffrement hybride est un sujet d'actualité qui a été proposé pour résoudre le problème de l'expansion du chiffré mais aussi pour montrer que le chiffrement homomorphe peut être employé dans des applications pratiques "réelles" autres que les applications "preuves de concept".
Le premier travail [GHS12] a proposé d'évaluer AES et a montré des résultats très grands en termes de ressources et de durées. Des travaux plus récents proposent d'évaluer Simon plutôt que l'AES, c'est-à-dire un schéma de chiffrement par blocs léger conçu pour être efficace sur architecture matérielle. En effet, du fait des contraintes actuelles des schémas homomorphes, cette approche est susceptible d'avoir un fort impact sur l'efficacité des évaluations. Nous résumons les résultats d'évaluation homomorphe de l'AES et de Simon dans le tableau II.8. Il s'avèrera plus tard que considérer Prince pourrait être un choix encore plus judicieux [DSES14].

Tableau II.8. Evaluation homomorphe de circuits connus sur des machines ayant des caractéristiques comparables

Année	Circuit	Schéma	Basé sur	λ*	# slots	Temps d'évaluation	Temps par bloc
2012	AES	BGV	RLWE	128	54	34 h	37 min
2012	AES	BGV	RLWE	128	720	65 h	5 min
2014	AES	CLT	Entiers	72	569	3 h 35 min	23 s
2014	AES	CLT	Entiers	80	1875	102 h	3 min 15 s
2014	SIMON	FV	RLWE	80	1800	51 min	1,70 s
2014	SIMON	FV	RLWE	128	2048	3 h 27 min	3,06 s
2014	SIMON	Yashe	RLWE	80	1800	17 min	0,57 s
2014	SIMON	Yashe	RLWE	128	2048	1 h 08 min	2,04 s

* ces valeurs sont approximatives

Tous ces travaux commencent par transformer le schéma de chiffrement par blocs en un circuit évaluable par leur cryptosystème. Ils utilisent les cryptosystèmes homomorphes par niveaux et la technique de batching. Le batching implique que chaque chiffré contient plusieurs "slots".

A chaque slot est associé un message en clair. Les opérations homomorphes s'effectuent sur les slots en parallèle. Les auteurs utilisent cette technique pour accélérer l'évaluation des différents circuits. Par conséquent, les temps relatifs par bloc sont raisonnables, mais il faut attendre que l'évaluation totale soit finie pour avoir les résultats. Le tableau II.8 montre que les temps d'évaluation totaux sont beaucoup améliorés par rapport à la première évaluation homomorphe de l'AES qui a duré 36 heures, surtout avec les schémas homomorphes par niveaux à base du problème RLWE.

II.3. Discussions

Dans cette section, je vais évoquer quelques problèmes sous-jacents à l'implantation logicielle du chiffrement homomorphe.

II.3.1. Algorithmique sur les données chiffrées

Je commence par décrire les difficultés rencontrées lorsque l'on veut évaluer des algorithmes sur des données chiffrées. Il est en effet souvent nécessaire de les modifier afin qu'ils soient compatibles avec la procédure d'évaluation homomorphe.
Un des problèmes le plus courant dans ce contexte est la présence de conditions qui dépendent de données chiffrées. En effet, si on peut calculer des formules booléennes sur des bits chiffrés pour évaluer une condition, il est impossible d'utiliser le résultat (qui est chiffré) pour guider le comportement de l'algorithme. De plus, cette impossibilité semble inhérente à la nature même du chiffrement homomorphe. Si un programme a un comportement totalement différent selon une condition donnée (qui dépend des données chiffrées), et que l'on puisse publiquement orienter l'exécution du programme en fonction de la valeur de la condition, un adversaire n'aurait plus aucun mal à casser l'indistinguabilité à clairs choisis du schéma de chiffrement utilisé. Ainsi, il faut évaluer l'intégralité des branches du programme quand leur exécution peut dépendre des données chiffrées. De même, une condition d'arrêt de boucle ne peut dépendre des données chiffrées : le programme doit exécuter le nombre maximal possible de tours de boucle. Cela a pour conséquence directe d'augmenter la complexité des algorithmes évalués sur les données chiffrées. En effet, il est impossible d'exploiter la structure des données associées pour résoudre un problème plus vite : l'utilisateur exécutant l'algorithme ne doit rien apprendre sur les données en claires. Par exemple, trier une liste aléatoire ou une liste déjà triée doit prendre le même temps. Cela implique que la complexité des algorithmes doit être calculée dans le pire cas. En pratique, malgré l'amélioration remarquable des performances des implantations de schémas de chiffrement homomorphe, cela affecte fortement les temps d'exécution sur les données chiffrées. Dans [FSF+13], plusieurs exemples d'implantations d'algorithmes simples (tri, somme, FFT, …) dans le domaine chiffré sont exposés. Entre autres, une opération de tri de 10 valeurs ayant 8 bits nécessite 3240 additions et 2790 multiplications sur des opérandes de tailles très grandes (plusieurs millions de bits). Cela demande un temps d'exécution important et des ressources énormes.

Nous décrivons les deux pratiques les plus répandues pour contourner un tel problème et pour pouvoir évaluer des conditions (de type si ... alors ...) et des terminaisons de boucles dépendant des données chiffrées [FSF+13]. Par contre, cela vient au prix d'une perte d'efficacité non-négligeable. Ces questions ont également été abordées dans [CDS15], où les auteurs proposent un compilateur qui transforme un programme écrit pour des entrées en clair en un programme pouvant être exécuté sur des entrées chiffrées.

Evaluation de conditions :
Pour évaluer une condition de type si p alors q sinon r, où p est un prédicat sur des données chiffrées, nous évaluons simplement l'expression suivante :

$$p.q + (1 - p).r$$

Cette transformation ne fait plus fuir d'informations sur la branche du programme retournée en sortie. En effet, les deux branches sont exécutées. De plus, le prédicat p étant évalué sur des données chiffrées, le bit de résultat est également chiffré. C'est donc également le cas de 1 - p. Or la multiplication de données chiffrées avec d'autres valeurs - que ces dernières soient des constantes ou également des chiffrés - retourne un résultat chiffré. Seul le possesseur de la clé secrète peut maintenant avoir accès au résultat du calcul. Ceci ne révèle aucune information sur les données claires, à condition que le schéma soit IND-CPA.

Terminaison de boucles :
Il reste maintenant le problème de la terminaison de boucle quand on a une condition dépendante des données chiffrées. Ici, on cherche aussi à éviter que le comportement du programme évalué sur les données chiffrées diffère en fonction des entrées de celui-ci. On peut par exemple voir directement les conséquences d'un programme composé d'une boucle while $c \neq Enc(0)$: il suffirait à un adversaire de surveiller juste le temps d'exécution pour connaître la valeur de c. Nous cherchons donc à avoir des boucles ayant des tailles indépendantes des données chiffrées afin de garantir un temps d'exécution constant. Pour ce faire, nous évaluons le nombre de tours de boucles maximal pire cas en clair, et nous rendons la boucle stationnaire dès lors que la condition de terminaison est remplie.

II.3.2. Performances des implantations logicielles

Les implantations logicielles de chiffrement homomorphe sont aujourd'hui nombreuses (tableau II.9). Mises à part les bibliothèques que nous avons présentées, peu sont publiques. Ces travaux ont permis de réduire l'écart entre la théorie et la pratique et ont montré que la cryptographie homomorphe existe en dehors des développements mathématiques et des fondements théoriques. Elle peut même être utilisée pour évaluer des circuits complexes comme l'AES.
Par contre, les résultats publiés montrent qu'une amélioration de l'efficacité de ces systèmes est encore nécessaire. Ils imposent des clés très longues, des données chiffrées prenant beaucoup plus de place que les données en clair, et les algorithmes de traitement sont coûteux en temps.
Une solution qui peut répondre à ce besoin est de passer à des plateformes alternatives telles que les GPUs, FPGAs ou encore ASICs pour implanter les cryptosystèmes homomorphes.

Tableau II.9. Les implantations logicielles de chiffrement homomorphe les plus connues

Schéma	Basé sur	CH	Référence	Publique
Gentry09	réseaux	total	[GH11b]	Non
vDGHV10*	entiers	total	[vDGHV10]	Non
SV11	réseaux	total	[PBS11b]	Oui
BGV12	RLWE	par niveaux	[GHS12]	Non
BGV12	R/LWE	par niveaux	[AMFF+13]	Non
FV12	RLWE	par niveaux	[LN14]	Oui
Yashe13	RLWE	par niveaux	[BLLN13]	Non
Yashe13	RLWE	par niveaux	[LN14]	Oui

* plusieurs mises à jour : [CMNT11, CNT12, CCK+13, CLT14]

II.4. Implantations alternatives aux processeurs généralistes

La majorité des implantations alternatives (sur GPU ou matérielles) dédiées à la cryptographie homomorphe a été consacrée à une ou plusieurs primitives les construisant. Très peu de travaux ont implanté un schéma homomorphe complet.

Nous nous focaliserons dans cette section plutôt sur les implantations matérielles des schémas homomorphes à base des problèmes (R)LWE vu qu'ils sont les plus efficaces d'un point de vue théorique. Les résultats des implantations logicielles et des évaluations homomorphes des circuits de chiffrement par blocs l'ont confirmé aussi.

II.4.1. Implantations dédiées aux schémas à base de réseaux et d'entiers

Les implantations pour les schémas à base de réseaux sont orientées vers le cryptosystème de Gentry et Halevi [GH11]. La plus ancienne [WYL+12] implante sur un GPU la multiplication modulaire composée d'une multiplication sur des grands entiers (opérandes de taille 1 million de bits environ) et d'une réduction modulaire. Le multiplieur modulaire a été ensuite utilisé pour implanter les primitives de chiffrement et de déchiffrement et l'opération de "recrypt". Le travail le plus récent de [DOS15DOS15] propose une amélioration de [WYL+12] sur un ASIC. Il correspond à la première implantation matérielle d'un schéma totalement homomorphe (sans la génération des clés). J'identifie dans le tableau II.10 les implantations dédiées au cryptosystème de Gentry et Halevi.

II.4.2. Implantations des schémas à base du problème RLWE

Bien que les cryptosystèmes basés sur le problème RLWE montrent plus d'efficacité, peu d'implantations de ces schémas sont présentées dans l'état de l'art en les comparant avec les schémas à base d'entiers. Les implantations matérielles dédiées aux schémas basés sur RLWE sont majoritairement consacrées à l'opération de multiplication polynomiale et sont conçues sur cibles FPGA. Dans le tableau II.11, un bon nombre de ces travaux sont listés.

Nous allons commenter maintenant quelques unes de ces contributions, et des choix algorithmiques et architecturaux adoptés.

Tableau II.10. Implantations dédiées au schéma de Gentry et Halevi

Référence	Plateforme	Observations
[WYL+12]	NVIDIA C2050 GPU	Multiplication : FFT Réduction modulaire : Algorithme de Barrett Opérandes ~ 1 million de bits Implantation de chiffrement, déchiffrement et "recrypt"
[WX13]	Stratix-V	Multiplication : FFT Opérandes ~ 786,000 bit
[WXN+14]	ASIC (90 nm)	Multiplication : FFT Opérandes ~ 786,000 bit
[DOS15]	ASIC (90 nm)	Multiplication : Algorithme de Shönhage Strassen Réduction modulaire : Algorithme de Barrett Opérandes ~ 1 million de bits Première réalisation d'un FHE en matériel

Tableau II.11. Implantations matérielles dédiées aux schémas à base de RLWE

Réf.	Fonctions implantées	Algorithme	Paramètres	Observations
(a)	multiplication polynomiale changement de modules relinéarisation	NTT	$n = 32768$ $log_2 q = 32$	accélérer l'évaluation homomorphe réalisée en logiciel
(b)	multiplication polynomiale	FFT NWC	$n \in [156, 2048]$ $log_2 q \in [20, 30]$	conception sur FPGA
(c)	multiplication polynomiale	Karatsuba FFT SchoolBook	$n = 512$ $log_2 q = 32$	conception sur FPGA
(d)	multiplication polynomiale réduction modulaire multiplication entière additions/soustractions sur des polynômes	NTT	$n = 2^{15}$ $log_2 q = 1128$	accélérer Yashe
(e)	multiplication polynomiale réduction modulaire	NTT	$n = 16384$ $log_2 q = 512$	accélérer les fonctions d'évaluation homomorphe
(f)	multiplication polynomiale	Karatsuba	$n \in [2500, 5120]$ $log_2 q \in [125, 230]$	accélérer la multiplication homomorphe et la relinéarisation de FV

(a) [DOS+15] (b) [DNF+15] (c) [JCM+15] (d) [RJV+15]
(e) [PNP+15] (f) [MRL+16b]

Dans [DOS+15], Doröz et al. ont proposé une architecture pour accélérer le cryptosystème homomorphe LTV basé sur le schéma NTRU. Cette architecture correspond à une implantation matérielle des opérations de multiplication polynomiale, changement de modules et relinéarisation afin d'accélérer la partie évaluation homomorphe réalisée en logiciel. Ils ont évalué un multiplieur sur des polynômes de degré 2^{15} avec des coefficients de taille 32 bits en utilisant l'algorithme de multiplication NTT et l'algorithme de réduction modulaire de Barrett. Le cœur de l'architecture est une unité arithmétique permettant de réaliser des opérations sur des nombres de 32 bits : addition et soustraction modulaire et multiplication sur des entiers. Les multiplications et accumulations des coefficients au niveau de la NTT sont effectuées en parallèle. Les coefficients des polynômes et les constantes de la NTT sont stockés dans des mémoires embarquées (BRAM). Les auteurs de [DOS+15] ont proposé de transformer les polynômes en entrée en un ensemble de polynômes de même degré avec des coefficients de taille plus petite en utilisant CRT. Ainsi, ils peuvent évaluer des polynômes avec des coefficients de taille plus grande que 32 bits. Les transformations CRT des coefficients de chaque polynôme sont traitées dans la partie logicielle. Les coefficients en représentation CRT sont ensuite envoyés au FPGA qui effectue la multiplication polynomiale. Dans ce travail, un intérêt particulier est porté au transfert de données entre la partie logicielle et la partie matérielle. Un bus PCI express est utilisé pour les transferts des polynômes d'entrée (opérandes) et de sortie (résultat) et des constantes NTT entre le FPGA et le PC. Les auteurs évaluent le coût de ces transactions pour un PCIe avec 8 voies capables de transporter 8 Gbit/sec.

Dans un deuxième travail [DNF+15], Chen et al. ont présenté une implantation matérielle de la multiplication polynomiale dédiée aux RLWE et SHE pour plusieurs valeurs de n et q.
Ils exploitent la FFT et NWC pour la conception de l'architecture du multiplieur. Un circuit pipeliné et une géométrie du flot de données constante sont deux caractéristiques importantes de l'architecture permettant d'exploiter le parallélisme et de réduire la latence des calculs. Chen et al. ont analysé et proposé une méthode de sélection de paramètres (n, q) tout en prenant en considération les exigences en sécurité et l'efficacité pour les systèmes basés sur le problème RLWE et SHE.

Les auteurs de [JCM+15] ont aussi étudié trois algorithmes de multiplication polynomiale (Karatsuba, FFT et SchoolBook) et ont proposé une architecture optimisée pour chacun des trois. Dans le but d'obtenir le résultat final avec une latence faible, ils ont comparé ces algorithmes pour des polynômes de degré 512 avec des coefficients de taille 32 bits tout en exploitation le parallélisme des calculs et en tenant compte de la consommation des ressources.
Ils ont aussi proposé une stratégie d'ordonnancement des opérandes (les coefficients des polynômes d'entrée) et ont conclu que l'algorithme classique SchoolBook est le plus efficace.

Roy et al. [RJV+15] présentent quant à eux une architecture des blocs nécessaires pour la construction du cryptosystème homomorphe Yashe avec $n = 2^{15}$ et un module $log_2 q = 1128\ bits$. Ils n'imposent pas de contraintes sur le choix du polynôme irréductible $f(x)$. Roy et al. ne prennent pas en considération le coût des transferts entre la mémoire externe et le FPGA et supposent qu'ils ont une bande passante de mémoire illimitée. Les blocs implantés

correspondent à une unité de calculs arithmétiques sur les polynômes (PAU), une unité pour le calcul de CRT et une unité de division et arrondissement (DRU). L'unité PAU implante une multiplication polynomiale basée sur NTT, la réduction modulaire de Barrett, une multiplication entière et les opérations d'additions/soustractions sur les polynômes. Les auteurs exploitent le parallélisme dû à la représentation en résidus des polynômes et introduisent deux notions de parallélisme vertical et parallélisme horizontal des calculs. L'architecture proposée est basée sur un flot de données simple et une architecture pipelinée qui nécessitent des améliorations vu certaines limitations (apparition de bulles dans le pipeline, schéma d'accès à la mémoire, etc.)

Dans un travail similaire [PNP+15], Pöppelmann et al. proposent une implantation matérielle des fonctions d'évaluation homomorphe (addition, multiplication) des cryptosystèmes basés sur le problème RLWE. Ils utilisent une plateforme de Catapult munie d'un FPGA Stratix V, deux modules mémoires de 4 GB chacun et une connexion PCI express. Le choix des paramètres est fixé à $n = 16384$ et $\lceil log_2\ p \rceil = 512$. Ils ont adopté l'algorithme de multiplication polynomiale à base de NTT et une réduction modulaire utilisant les nombres premiers de Solinas. Un multiplieur sur des grands entiers est également inclus dans l'architecture mais limité à des tailles de 576×576 bits. Ils n'ont pas pu implanter un multiplieur pipeliné sur des données 1040×1040 bits car les ressources disponibles sur le FPGA ne sont pas suffisantes. Comme la plateforme Catapult dispose d'une mémoire externe, Pöppelmann et al. ont proposé un mécanisme d'accès à la mémoire basé sur un double tampon.

Dans une suite de travaux [MRL+15, MRL+16a et MRL+16b], Migliore et al. se sont intéressés à l'accélération matérielle de schémas à base de RLWE et plus particulièrement à la multiplication polynomiale. Dans [MRL+15], ils ont proposé une architecture permettant d'accélérer la multiplication polynomiale et ont étudié l'effet de cette architecture sur les algorithmes de Karatsuba et FFT. Aucun résultat de synthèse n'est donné dans ce travail mais une étude visant à optimiser le temps des calculs est présentée. Migliore et al. ont conclu que l'algorithme de Karatsuba est plus efficace que FFT (moins de transfert de données, étapes de pré- et post-calculs plus simples, dépendance faible de données et donc une possibilité de parallélisation). Ils ont proposé une stratégie pour implanter Karatsuba en matériel en concevant un petit multiplieur de degré p et en construisant de manière récursive le polynôme de sortie. Ils ont donné une estimation du nombre d'opérations nécessaires pour finaliser la multiplication polynomiale et ont étudié plusieurs stratégies d'ordonnancement des coefficients des polynômes d'entrée et sortie afin de diminuer le temps de calcul. Dans [MRL+16a] et la version plus récente [MRL+16b], ils ont proposé une conception logicielle/matérielle visant à accélérer les deux opérations de multiplication homomorphe et de relinéarisation du schéma FV. La multiplication polynomiale est implantée en matériel avec l'algorithme de Karatsuba. Le processeur embarque les deux opérations du schéma FV et communique avec le FPGA à travers un bus PCI express. Le choix des paramètres est fixé à $n \in [2500, 5120]$ et $log_2 q \in [125, 230]$.

II.5. Bilan et Synthèse

Après avoir décrit les implantations matérielles dédiées au chiffrement homomorphe, je résume dans la suite les points les plus importants à retenir :

- Les implantations matérielles dédiées au chiffrement homomorphe basé sur RLWE prennent de plus en plus d'importance ces dernières années. Des facteurs d'accélération allant de 10 jusqu'à 100 en moyenne sont notés dans l'état de l'art en comparaison avec des implantations logicielles. Ainsi, nous pouvons confirmer que le recours aux plateformes alternatives offre des améliorations notables des performances des schémas homomorphes surtout au niveau des durées de calcul.
- Etant la plus coûteuse et la plus utilisée, la multiplication polynomiale est l'opération implantée en matériel dans tous les travaux existants. Ces derniers font appel majoritairement à l'algorithme FFT et sa version spécifique NWC vu qu'ils ont une complexité asymptotique faible. Par contre, ils imposent des étapes de pré et post-calculs et des paramètres supplémentaires à considérer. En plus, la NWC empêche l'utilisation du batching, qui permet de mettre au sein d'un même chiffré plusieurs messages en parallèle et est particulièrement recommandé car il permet de réduire fortement le temps de calcul par message.
  Le choix de l'algorithme de multiplication est un élément à considérer en tenant compte des contraintes exposées mais aussi des ressources matérielles disponibles et des possibilités d'accélération de chaque algorithme.
- Certains travaux de la littérature ont proposé des conceptions logicielles/matérielles visant à accélérer une ou plusieurs primitives homomorphes. La communication entre le processeur et le FPGA est un point très important et a un impact fort sur les performances de l'accélérateur proposé compte tenu de la taille des données manipulées.
- La taille des données est aussi un point à étudier davantage lorsqu'on parle d'implantations matérielles vu que les ressources mémoires d'un FPGA sont limitées.
- Comme nous l'avons déjà vu au premier chapitre, l'extraction de paramètres pratiques pour les schémas homomorphes à base de RLWE est un sujet d'actualité qui doit être pris en compte si on veut viser des applications réelles et réduire l'écart entre la théorie et la pratique. Néanmoins, les configurations de l'état de l'art sont limité à des paramètres fixes de tailles plus au moins petites. Et toutes les optimisations et les améliorations sont proposées dans le cadre de cette configuration. Des architectures flexibles visant des paramètres pratiques et plus grands deviennent nécessaires.

II.6. Paramètres pratiques pour FV et Yashe

Dans cette section, je reviens sur un aspect pratique de l'extraction des paramètres pour des schémas à base de RLWE évoqué dans la section I.4.5. Je donne un exemple de paramètres relatifs aux schémas de Yashe et FV extraits des travaux de [LN14]. Le travail de Lepoint et Naherig repose sur une amélioration de l'approche de Van de Pol et Smart et de la version complète de l'article de BKZ-2.0.

Je résume dans les tableaux II.12 et II.13 les valeurs minimales de $log_2(q)$ pour différents n et L. Ces valeurs sont extraites pour un écart type $\sigma = 8$, un module $t = 2$ et une base $w = 2^{32}$.
Pour pouvoir évaluer 10 multiplications homomorphes sur des données chiffrées par Yashe dans un anneau de polynômes de dimension n=8192, il faut prendre un module q tel que $log_2(q) = 326$. Ces paramètres sont loin d'être comparables aux configurations évoquées dans les implantations matérielles de l'état de l'art.
Une liste plus détaillée de configurations est donnée en annexe 1. Je regroupe dans cette liste plusieurs combinaisons possibles de n, q et L en jouant à chaque fois sur un ou plusieurs paramètres. Elle est extraite d'un ensemble de travaux récents et répond, sous certaines conditions, au compris efficacité-sécurité-fonctionnalité. Elle pourra guider le concepteur dans la définition de son cahier de charge en se basant sur les contraintes de son application.

Tableau II.12. Exemple de paramètres concrets pour le schéma Yashe
Valeur minimale de $log_2(q)$ pour plusieurs valeurs de n et de niveau de multiplications L

n	**16384**	**8192**	**4096**	**2048**	**1024**
L=0	24	23	22	21	20
L=1	70	68	66	64	62
L=10	346	326	306	286	265
L=50	1550	1450	1350	1250	1150

Tableau II.13. Exemple de paramètres concrets pour le schéma FV
Valeur minimale de $log_2(q)$ pour plusieurs valeurs de n et de niveau de multiplications L

n	**16384**	**8192**	**4096**	**2048**	**1024**
L=0	23	22	21	20	19
L=1	52	49	46	43	40
L=10	313	292	271	250	229
L=50	1473	1372	1271	1170	1069

II.7. Conclusion

Dans ce chapitre, nous avons cherché à présenter l'aspect pratique de la cryptographie homomorphe. Nous avons décrit les implantations logicielles et matérielles dédiées aux différentes familles de schémas avec un intérêt particulier porté aux schémas basés sur RLWE. Les réalisations logicielles ont montré que le chiffrement homomorphe peut bien être déployé en pratique et des exemples réels ont été donnés. Par contre, une amélioration des performances est nécessaire, ouvrant la voie à l'utilisation de plateformes alternatives (GPU, FPGA et ASIC). Dans le cas des implantations matérielles, nous avons noté des facteurs d'accélération importants dans l'état de l'art qui confirment l'intérêt de cette direction.

Nous avons également mentionné plusieurs points qui doivent être pris en compte si nous voulons proposer une architecture qui peut être utilisée dans des applications pratiques. C'est sur ces points que nous allons nous baser pour présenter notre première contribution qui est la génération d'un multiplieur polynomial de grande taille et flexible au chapitre III.

Chapitre III. Génération d'un multiplieur polynomial de grande taille et flexible

Dans ce chapitre, je décrirai le multiplieur polynomial modulaire développé afin d'améliorer les performances des schémas homomorphes basés sur RLWE. La multiplication polynomiale modulaire ayant fait l'objet de plusieurs travaux, je consacrerai les trois premières sections à aborder les différents axes de l'étude. Je commencerai par présenter un prototype du schéma BGV réalisé avec Sage qui sera ensuite profilé, avec d'autres implantations, en vue d'identifier les opérations les plus coûteuses et les plus utilisées dans la construction des cryptosystèmes basés sur RLWE. J'analyserai, par la suite, l'opération la plus critique, à savoir la multiplication polynomiale modulaire. Dans la deuxième partie de ce chapitre, je présenterai l'accélérateur développé en justifiant les choix algorithmiques et architecturaux. Les sections III.6 et III.7 seront consacrées à la description des implantations, l'analyse des résultats et la proposition d'optimisations possibles.

III.1. Prototypage du schéma BGV avec Sage

J'ai réalisé au début de la thèse une implantation du schéma BGV avec Sage [S+17]. Ce travail m'a permis de bien comprendre l'enchaînement des calculs dans la construction d'un tel schéma et d'évaluer les performances des différentes primitives qui le définissent. Cette implantation servira également de référence pour la comparaison avec les implantations matérielles que je détaillerai dans les sections III.6 et IV.2.

Nous avons montré au premier chapitre que les schémas homomorphes les plus efficaces à ce jour sont ceux de la seconde génération. Etant les plus récents et faisant appel à la technique d'invariance à l'échelle, FV et Yashe ont pris le devant. Jusqu'à récemment, Yashe était considéré comme le plus prometteur vu la taille relativement réduite des clés et des chiffrés. Cependant, de nouvelles attaques basées sur l'attaque "subfield lattice attack" introduite dans [ABD16] affaiblissent sa sécurité et nous amènent à considérer FV quoique moins efficace. En plus, Costache et Smart ont mené une étude [CS15a] afin de répondre à la question "Quel schéma homomorphe à base de RLWE est le meilleur ?" et ont conclu que BGV est plus efficace pour des modules en clair de grande taille :

"*We find that the BGV scheme appears to be more efficient for large plaintext moduli, whilst YASHE seems more efficient for small plaintext moduli (although the benefit is not as great as one would have expected). … which is the most efficient scheme depends on the context (message characteristic and depth of admissible circuits)*" [CS15a]

Ainsi, on peut dire que notre implantation demeure pertinente et qu'elle peut servir pour construire aussi bien Yashe que FV car ces schémas utilisent majoritairement les mêmes primitives.

Pour réaliser l'implantation et les tests, j'ai utilisé Sage Mathematics Software [S+17] dans sa version 6.6. J'ai exploité les outils relatifs à l'algèbre et à la théorie des nombres permettant de manipuler les anneaux, les polynômes, les corps finis ainsi que les corps cyclotomiques.

Sage se veut une alternative libre à l'ensemble des logiciels propriétaires (Magma, Maple, Mathematica et Matlab). L'interface, écrite en python, permet d'utiliser conjointement les capacités de calcul de ses différents paquetages. Sage permet de faire des mathématiques générales et avancées, pures et appliquées. Il couvre une vaste gamme de mathématiques, dont l'algèbre, l'analyse, la théorie des nombres, la cryptographie, l'analyse numérique, l'algèbre commutative, la théorie des groupes, la combinatoire, la théorie des graphes, l'algèbre linéaire formelle, etc [S+17]. Cela nous a orientés vers cette plateforme libre de calculs mathématiques pour implanter le schéma BGV.

Aperçu général :
Pour construire le prototype logiciel du schéma BGV, j'ai classé les fonctions nécessaires à sa construction en trois grandes catégories :

- Les fonctions élémentaires correspondent aux fonctions basiques que je vais utiliser pour définir les fonctions de la deuxième catégorie,
- Les fonctions du chiffrement homomorphe sont la génération des clés, le chiffrement, le déchiffrement, l'addition homomorphe et la multiplication homomorphe,
- Les fonctions de test déterminent le nombre d'opérations que le schéma peut supporter ainsi que la durée des calculs.

Quelques résultats :
Je résume dans les tableaux III.1 et III.2 quelques résultats de notre prototype du schéma BGV.
La description du cryptosystème avec Sage permet une modification rapide des paramètres offrant ainsi la possibilité d'étudier le comportement du schéma (taille des données et croissance du bruit) en fonction de ces paramètres.

Tableau III.1. Profondeur multiplicative en fonction de la dimension n

Dimension n	Profondeur multiplicative
2048	3
4096	5
8192	11

Tableau III.2. Taille des clés en fonction de la dimension n

Dimension n	Taille de sk (KB)	Taille de pk (KB)
2048	19	40
4096	58	115
8192	290	580

III.2. Profilage d'implantations logicielles de schémas homomorphes basés sur RLWE

Dans le but de déterminer l'opération arithmétique la plus coûteuse dans la construction des schémas homomorphes à base du problème RLWE, nous avons effectué une analyse et un profilage de quelques implantations logicielles, à savoir notre prototype avec Sage et les deux implantations de FV et Yashe de Lepoint [Lep14].

Notre analyse a montré que les opérations les plus critiques sont la multiplication polynomiale, la réduction modulaire et l'échantillonnage. Des opérations comme l'addition polynomiale ou l'addition et la multiplication d'entiers sont fréquentes mais nécessitent un temps d'exécution faible.

Les résultats du profilage illustrés par la figure III.1 montrent que la multiplication polynomiale consomme à elle seule entre 41 et 58% du temps d'exécution global.

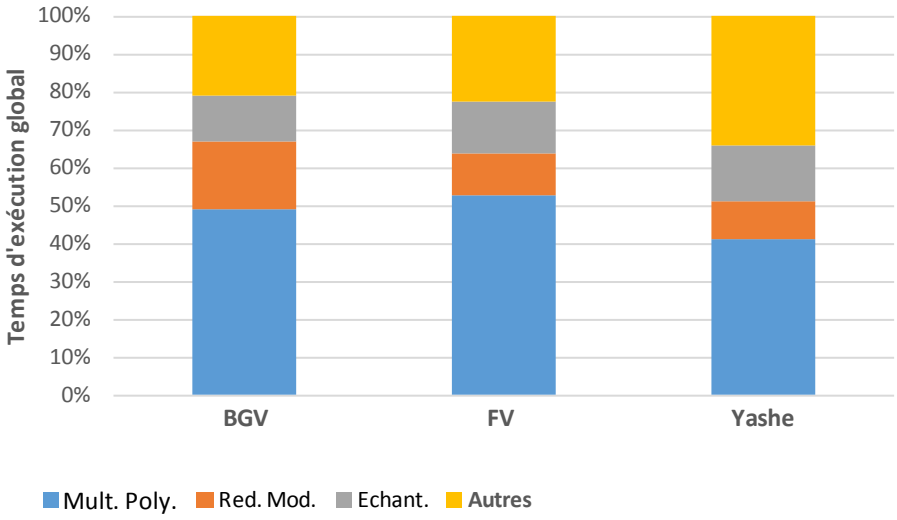

Figure III.1. Taux d'exécution des opérations les plus critiques

De plus, cette opération intervient dans les primitives de génération des clés, de chiffrement, de déchiffrement et de multiplication homomorphe (seule l'addition homomorphe n'y fait pas appel), même si le nombre d'occurrences varie d'une primitive à une autre et d'un schéma à un autre. Par exemple, une multiplication homomorphe évaluée par Yashe requiert 1 multiplication polynomiale et 1 relinéarisation alors que celle évaluée par FV nécessite 4 multiplications polynomiales et 2 relinéarisations.

III.3. La multiplication polynomiale modulaire dans les schémas RLWE

Le problème RLWE est construit autour d'un anneau de polynômes de la forme $R_q = \frac{\mathbb{Z}_q[X]}{f(x)}$ avec $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ et f(x) un polynôme irréductible dans $\mathbb{Z}_q$ de degré n. n définit également la dimension de l'anneau R_q.

III.3.1. Description

Nous décrivons dans cette section la multiplication polynomiale modulaire dans $R_q = \frac{\mathbb{Z}_q[X]}{f(x)}$.
Soit $A(x) = a_{n-1}x^{n-1} + \ldots . + a_0 \equiv (a_{n-1}, \ldots . , a_0)$ et $B(x) = b_{n-1}x^{n-1} + \ldots . + b_0 \equiv (b_{n-1}, \ldots . , b_0)$ deux polynômes de R_q. Les coefficients des polynômes de R_q sont des éléments de $\mathbb{Z}_q$, satisfaisant ainsi $0 \leq a_i,\ b_i < q$ avec $i \in [\![0, n-1]\!]$.
Calculer $C(x) = A(x) \times B(x)$ dans R_q revient à calculer le produit terme à terme de $A(x)$ *et* $B(x)$ puis réduire le résultat modulo $(q, f(x))$.

Exemple :
Illustrons cette opération par un exemple. Pour ce faire, prenons $n = 4,\ q = 5$ et $f(x) = x^4 + 1$.
Soit $A(x) = x^3 + 3x^2 + 4x^1 + 1 \equiv (1,3,4,1)$
Et $B(x) = 2x^3 + 1x^2 + 4x^1 + 0 \equiv (2,1,4,0)$
Comme le montre la figure III.2, le résultat $C(x) = A(x) \times B(x)$ mod$(q, f(x))$ est donné par $C_{dans\ R}$ (3,0,2,0).

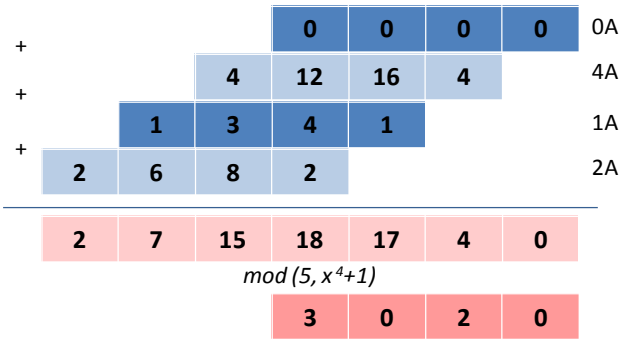

Figure III.2. Exemple de multiplication polynomiale modulaire dans $\boldsymbol{R_q}$

III.3.2. Paramètres mis en jeu

L'opération de multiplication polynomiale modulaire met en jeu trois paramètres, à savoir le degré n, le module q et le polynôme irréductible f(x). n définit le degré des polynômes de R_q ; tout $P \in R_q$ est de degré (n-1). q représente la taille des coefficients de P qui correspond à $log_2(q)$.
Dans le contexte du chiffrement homomorphe, il est nécessaire de faire un dimensionnement au plus juste de l'application à exécuter. Comme nous l'avons déjà vu, le choix de n et q se fait en liaison étroite avec la profondeur multiplicative et le niveau de sécurité souhaités et le choix de f(x) dépend de la nécessité d'optimisations (batching ou SIMD) et de l'algorithme de multiplication employé. Ainsi, plus on a de flexibilité sur le choix de ces paramètres, plus on pourra couvrir un domaine d'applications plus large et autoriser des optimisations visant à améliorer les performances.

III.3.3. Algorithmes

Nous présentons une étude comparative des différents algorithmes de multiplication polynomiale dans le contexte du chiffrement homomorphe. Nous étudions les trois algorithmes les plus utilisés : l'algorithme classique (schoolbook), l'algorithme de Karatsuba ainsi que celui de la FFT. Dans ces algorithmes, trois étapes sont généralement nécessaires : Une étape de pré-traitement, une étape de traitement (multiplication des sous-produits) et une étape de post-traitement. Pour optimiser l'étape de traitement en réduisant le nombre de sous-produits à multiplier, il est faut adapter les étapes de pré et post-traitement.
La comparaison concerne l'opération de multiplication polynomiale uniquement sans la réduction modulaire (à part dans un cas particulier, cf. section III.3.3.3). La réduction modulaire est traitée dans la section III.4.

Notation :
Soit $A(x) = a_{n-1}x^{n-1} + \ldots . + a_0 \equiv (a_{n-1}, \ldots . , a_0)$ et $B(x) = b_{n-1}x^{n-1} + \ldots . + b_0 \equiv (b_{n-1}, \ldots . , b_0)$ deux polynômes de R_q.
Le but est de calculer $R(x) = A(x) \times B(x)$.

III.3.3.1. Algorithme de multiplication classique (schoolbook)

L'algorithme de multiplication classique est obtenu en développant le produit terme à terme de deux polynômes. Cela revient à calculer $R(x) = A(x) \times B(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j \, x^{i+j}$.
En effet, chaque élément du résultat est donné par :

$$R(i+j) = R(i+j) + A(i) \times B(j)$$

Un élément de rang k dans le résultat correspond à une multiplication avec une accumulation additive de tout élément de A et B dont la somme des indices i et j est égale à k.
Pour des polynômes de degré n-1, le nombre d'opérations élémentaires est $\mathcal{O}(n^2)$. Aucun pré ou post-traitement n'est nécessaire et aucun paramètre supplémentaire n'est requis.

III.3.3.2. Algorithme de Karatsuba

L'algorithme de Karatsuba représente un premier raffinement de l'algorithme de multiplication classique. Il permet de réduire le nombre de sous-produits à multiplier pendant l'étape de traitement. A cet effet, on divise les polynômes A et B en deux parties de même degré. $\frac{n}{2} - 1$. A_H et B_H représentent les polynômes composés des hauts degrés de A et B respectivement, tandis que A_L et B_L correspondent aux degrés les plus bas. Ainsi, on obtient :

$$A = A_L + A_H . x^{\frac{n}{2}} \quad \text{et} \quad B = B_L + B_H . x^{\frac{n}{2}}$$

Le résultat de la multiplication est donné par :

$$R = A \times B = \left(A_L + A_H . x^{\frac{n}{2}}\right) \times \left(B_L + B_H . x^{\frac{n}{2}}\right) = A_L B_L + (A_L B_H + A_H B_L) . x^{\frac{n}{2}} + A_H B_H x^n$$

$$= A_L B_L + [(A_H + A_L).(B_H + B_L) - A_H B_H - A_L B_L] . x^{\frac{n}{2}} + A_H B_H x^n$$

L'algorithme de Karatsuba nécessite 3 sous-multiplications au lieu de 4, en contrepartie d'une étape de pré-traitement (construction des A_L , A_H, B_L et B_H) et d'une étape de post- traitement pour la reconstruction du résultat final.
Pour des polynômes de degré n-1, la complexité de cet algorithme est $\mathcal{O}(n^{log_2 3}) \approx \mathcal{O}(n^{1,59})$.

III.3.3.3. Algorithme de la FFT

Les algorithmes à base de la transformée de Fourier rapide (appelée aussi FFT pour Fast Fourier Transform) sont ce que l'on sait faire de mieux pour multiplier des polynômes. La complexité de l'algorithme de la FFT est en $\mathcal{O}(n\ log_2 n)$. Il existe d'autres algorithmes à base de FFT tels que la NWC (Negative Wrapped convolution) ou l'algorithme de Schönhage-strassen.
Globalement, l'algorithme de la FFT correspond à une multiplication par interpolation, pour laquelle les points d'interpolation sont des racines k-ième de l'unité. Il fait intervenir des étapes de pré- et post-traitement complexes et nécessite des paramètres supplémentaires pour le calcul du résultat final. La multiplication de deux polynômes se fait en trois étapes : une opération de FFT sur chaque polynôme, une multiplication des coefficients résultants de la FFT et une IFFT (FFT inverse) pour construire le résultat final donné par :

$$A \times B = IFFT(FFT(A) \odot FFT(B))$$

où $\odot$ est la multiplication terme à terme des coefficients.

III.3.3.4. Comparaison des algorithmes de multiplication dans le contexte du chiffrement homomorphe

Nous revenons, dans cette section, sur les points à retenir des différents algorithmes de multiplication appliqués dans un scénario de chiffrement homomorphe :

- Pour l'algorithme de multiplication classique, aucun pré- et post- traitement particulier n'est nécessaire. Il n'y a aucune contrainte sur les paramètres n, q et f(x) autorisant ainsi l'utilisation du batching. Et les seules données à manipuler sont les coefficients des polynômes. Cependant, on a une complexité globale en $\mathcal{O}(n^2)$.
- L'algorithme de Karatsuba améliore la complexité en faisant appel à des étapes de pré et post-traitement. Pour garantir une efficacité meilleure, il exige que $n = 2^i\, p$ avec $(i, p) \in \mathbb{Z}^2$ [MRL+16b]. Karatsuba autorise aussi l'utilisation du batching.
- L'utilisation de la FFT dans un contexte de chiffrement homomorphe a été étudiée dans [DNF+15]. Pour une FFT de taille N, l'algorithme de la FFT réalise une multiplication polynomiale suivie d'une réduction modulo un polynôme de degré N. Ce dernier est généralement de la forme $x^N - 1$, qui n'est pas irréductible ; condition nécessaire du problème RLWE. Pour pallier ce problème, on peut soit prendre un $N \geq 2n$, soit prendre un polynôme de la forme $x^N + 1$. La deuxième solution est connue sous le nom du NWC. Elle divise par deux la largeur de la FFT requise et permet de réaliser la réduction modulaire polynomiale pendant la FFT. Dans le contexte du chiffrement homomorphe, un nombre de conditions est à respecter dans le cas de la NWC :
  - Le degré n doit être une puissance de 2
  - Le module q doit être premier et satisfaire $q \equiv 1\ mod\ 2n$
  - Il faut imposer $f(x) = x^n + 1$; cette condition représente l'inconvénient majeur de la NWC vu qu'elle interdit le batching.

En conclusion, nous pouvons dire que dans un scénario applicatif donné de chiffrement homomorphe et dans une optique de flexibilité et de généricité, l'algorithme classique se révèle une alternative intéressante. Il ne pose aucune contrainte sur le choix des paramètres, nous donnant ainsi la liberté de faire un paramétrage propre aux systèmes homomorphes à base du problème RLWE. Il autorise aussi l'utilisation du batching (Tableau III.3).

Un travail réalisé dans notre équipe [JCM+15] a comparé les trois algorithmes (FFT, Karatsuba, Schoolbook) dans un contexte d'accélération matérielle de l'opération de multiplication polynomiale pour des cryptosystèmes homomorphes. Il a conclu que l'algorithme classique convient parfaitement pour une implantation sur une cible matérielle de type FPGA car le taux de parallélisation est grand, le besoin en ressources est plus petit que la FFT et Karatsuba et la structure est simple. Par ailleurs, les éléments disponibles dans les architectures de FPGA sont bien adaptés pour optimiser les temps de calcul, réduisant le gain obtenu avec l'algorithme de Karatsuba, plus complexe à implanter et qui est de plus pénalisé par les pré- et post-traitements.

Tableau III.3. Comparaison des algorithmes de multiplication dans un contexte de chiffrement homomorphe (CH)

Algorithmes	Classique	Karatsuba	FFT/NWC
Complexité	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{1,59})$	$\mathcal{O}(n\ log_2 n)$
Contraintes sur n	Non	$n = 2^i\ p$ * $(i,p) \in \mathbb{Z}^2$	puissance de 2
Contraintes sur q	Non	Non	q premier et $q \equiv 1\ mod\ 2n$
Contrainte sur f(x)	Non	Non	$f(x) = x^n + 1$
Pré- et post-traitements	Non	Oui	Oui
Paramètres supplémentaires	Non	Oui	Oui
Optimisations relatives au CH	Oui	Oui	Non

* une meilleure efficacité [MRL+16b]

III.4. Spécification d'un multiplieur polynomial modulaire de grande taille et flexible

III.4.1. Objectifs

La littérature présente une diversité d'architectures matérielles et logicielles/matérielles visant à accélérer la multiplication polynomiale (modulaire) afin d'améliorer les performances des cryptosystèmes homomorphes. Une synthèse sur les travaux existants a été présentée dans la section II.5 et nous amène à spécifier les caractéristiques de notre multiplieur afin de répondre aux différentes exigences.

Ces caractéristiques se résument en quatre points principaux :

- Grand : notre multiplieur doit supporter des polynômes de tailles "très" grandes à la fois en degré et en taille des coefficients ;
- Flexible et générique : notre approche doit permettre au concepteur de configurer facilement les trois paramètres principaux (le degré n, la taille des coefficients q et le polynôme réducteur f(x)) ;

- Adaptable : le multiplieur doit s'adapter à n'importe quel schéma homomorphe à base du problème RLWE ;
- Evolutif : il doit prendre en compte les ressources de cibles matérielles variées et les contraintes en sécurité.

III.4.2. Choix algorithmiques et architecturaux

Pour répondre aux objectifs que nous venons de citer, nous proposons une conception avec 4 niveaux hiérarchiques (figure III.3), basés essentiellement sur la décomposition des polynômes opérandes en sous-polynômes ayant des degrés et des coefficients plus petits. Plus précisément, nous avons consacré les deux niveaux hauts (Niveau 1 et Niveau 2) aux grandes opérandes où sera effectué le paramétrage du degré n et de la taille de coefficients $log_2 q$. Les deux niveaux plus bas (Niveau 3 et Niveau 4), quant à eux, concernent les sous-polynômes. Ces derniers arrivent après décomposition et ont des degrés et tailles des coefficients plus petits. C'est aux niveaux 3 et 4 que nous effectuons la multiplication polynomiale composée de produits terme à terme des coefficients et d'accumulations successives.

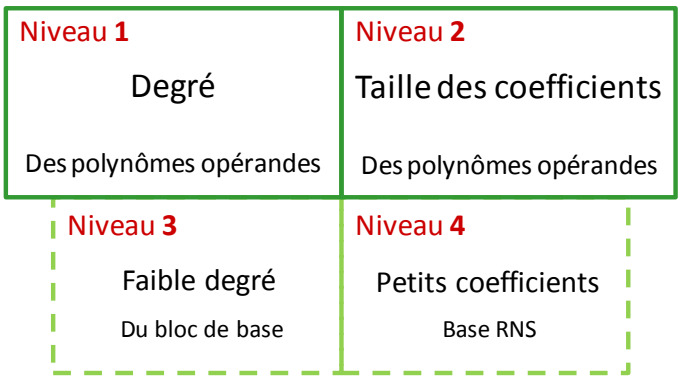

Figure III.3. Aperçu général de l'architecture proposée avec 4 niveaux hiérarchiques

Le flot de calcul, décrit dans l'algorithme 1, peut être divisé en 3 grandes étapes : premièrement la décomposition des polynômes, ensuite la multiplication polynomiale et enfin la réduction modulaire polynomiale. L'étape de la décomposition des polynômes sera détaillée dans les sections III.4.2.1 et III.4.2.2. Pour réaliser la multiplication polynomiale, nous avons choisi l'algorithme de multiplication classique schoolbook qui se ramène à des multiplications modulaires entières (entre les coefficients) plus des accumulations successives. A cette étape, pour deux polynômes de degré K et de taille des coefficients $log_2 q$, le polynôme résultant a les caractéristiques suivantes :

- Degré du polynôme : $(K-1) \times 2 + 1$
- Taille des coefficients : $log_2 q$

Une fois la multiplication des différents sous-polynômes terminée, une étape de reconstruction produit le polynôme noté C'(x) à partir des différents sous polynômes résultats. C'(x) subit une réduction modulo le polynôme réducteur f(x). Il en résulte le polynôme C(x). Comme nous n'avons fixé aucune contrainte sur la forme de f(x) à part qu'il soit irréductible, nous avons implanté une version basée sur l'algorithme de la division euclidienne des polynômes.

Algorithme 1: Multiplieur polynomial modulaire de grande taille et flexible

Entrée: Deux polynômes de degree n: $A(x)$et $B(x)$
Degré du bloc de base K
Coeff_L, Coeff_S, q
Polynôme réducteur f(x)

Sortie: C(x) = A(x) × B(x) mod (q, f(x))

1. Nombre des sous-polynômes: $nb \leftarrow \left\lfloor \frac{n+1}{K} \right\rfloor + 1$

2. Décomposition de $A(x) = \{A_1(x), \dots, A_{nb}(x)\}$

3. Décomposition de $B(x) = \{B_1(x), \dots, B_{nb}(x)\}$

4. $T \leftarrow \left\lfloor \frac{Coeff\_L}{Coeff\_S} \right\rfloor + 1$ // dimension de la base RNS

5. $\mathcal{B}_{RNS} = \{p_1, p_2, \dots, p_T\}$
avec $(PGCD\,(p_i, p_j)_{i \neq j} = 1\,)_{i=1..T}^{j=1..T}$ et $|p_i| = Coeff\_S$ bits

6. **Pour** i **allant de** 1 **à** nb **faire**
  // Chargement de $A_i(x)$
  RNS $(a_{i,0}, \dots, a_{i,K})$
  Pour j **allant de** 1 **à** nb **faire**
    //Chargement de $B_j(x)$
    RNS $(b_{j,0}, \dots, b_{j,K})$
    $T(x) = \sum_{l=0}^{K} \sum_{m=0}^{K} IRNS\,(RNS(a_{i,l}) \times RNS(b_{j,m})\ mod\ q)$
    //Envoi de $T(x)$

7. Reconstruire $C'(x)$ des différents $T(x)$

8. $C(x) = C'(x)\ mod\ f(x)$

9. Retourner $C(x)$

III.4.2.1. La décomposition appliquée aux polynômes

Dans un contexte de chiffrement homomorphe, les polynômes que nous manipulons sont très grands à la fois en degré et en taille des coefficients. Comme notre multiplieur vise à être générique et flexible, la décomposition des polynômes opérandes en des sous-polynômes plus petits nous permet de gérer n'importe quelles tailles tout en tenant compte des ressources disponibles sur notre cible matérielle.

Pour simplifier la compréhension, nous allons commencer par décrire la décomposition appliquée aux polynômes et passer, dans la section suivante, aux coefficients.

Durant l'étape de décomposition, les polynômes opérandes A(x) et B(x) vont être transformés en deux ensembles $\{A_1(x), \dots, A_{nb}(x)\}$ et $\{B_1(x), \dots, B_{nb}(x)\}$ de sous-polynômes de degré fixe que nous noterons K. Ensuite, chaque polynôme du premier ensemble va être multiplié avec tous les polynômes du second ensemble, ce qui revient à une multiplication (polynomiale) terme à terme entre deux ensembles. L'algorithme appliqué aux $A_i(x)$ et $B_j(x)$ est le même, il implante la méthode schoolbook sur des polynômes de taille fixe. Il sera modélisé par un bloc noté *Polynome_K* .

Le choix de la valeur de K est fixé au début des calculs en fonction des ressources de la cible matérielle (en particulier la quantité de mémoire) mais aussi d'autres critères que nous détaillerons dans la section III.6.

Exemple :
J'introduis un exemple où je fixe $K = 3$ et je prends $n = 8$ (des valeurs très petites juste pour l'illustration, pour des raisons de clarté).
Soit $A(x) = a_8x^8 + a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0$
$\equiv (a_8, a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$
Et $B(x) = b_8x^8 + b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$
$\equiv (b_8, b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$
Je décompose $A(x)$ et $B(x)$ en des sous polynômes de degré $K = 3$ comme illustré en figure III.4.

$A(x)$	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
	$A_3(x)$			$A_2(x)$			$A_1(x)$		

$B(x)$	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
	$B_3(x)$			$B_2(x)$			$B_1(x)$		

Figure III.4. Exemple de décomposition des polynômes

J'utilise le bloc *Polynome_K* pour faire les multiplications sur les $A_i(x)$ et $B_j(x)$. Dans cet exemple, *Polynome_K* va être appelé 9 fois. Une fois les calculs terminés, je reconstruis $C'(x)$. Cette reconstruction est basée sur un ré-ordonnancement des coefficients des polynômes résultants du bloc *Polynome_K* et des additions de ces coefficients.

III.4.2.2. La décomposition appliquée aux coefficients

Etant donné que les coefficients sont de grands entiers, nous avons choisi la représentation modulaire des nombres RNS pour les décomposer en des entiers plus petits.

Représentation modulaire des nombres RNS : [BM04] [Big14]
La représentation modulaire des nombres RNS (Residue Number System) est une généralisation du théorème des restes chinois. Elle permet de construire un système de représentation non positionnel, dans lequel un entier est représenté par ses restes modulaires.

Théorème : Théorème des Restes Chinois :
Soient $(m_1, \ldots, m_m)$ un ensemble de nombres strictement positifs deux à deux premiers entre eux et $M = \prod_{i=1}^{m} m_i$. Soit $(x_1, \ldots, x_m)$ un m-uplet d'entiers tel que $x_i < m_i$.
Il existe un unique X qui vérifie :

$$0 \leq X < M \quad et \quad x_i = X \bmod m_i \ \ pour \ \ 1 \leq i \leq m$$

Cette représentation permet d'effectuer des calculs rapides sur de très grands nombres.

Dans cette représentation, les entiers de l bits sont découpés en m morceaux bien plus petits, de w bits seulement (avec $m \times w \geq l$).
En RNS, un entier X sera représenté par $X_{RNS} = (x_1, \ldots., x_m) = (X \bmod m_1, \ldots., X \bmod m_m)$ où $(m_1, \ldots., m_m)$ sont des modules de w bits deux à deux premiers entre eux.
On dit que $(m_1, \ldots., m_m)$ est la base RNS et X_{RNS} est la représentation RNS de X.
On appelle canal tous les traitements effectués modulo un des éléments de la base.
L'originalité du RNS vient du fait que pour un certain nombre d'opérations (addition, soustraction, multiplication …) les calculs sont faits indépendamment sur chaque canal, sans propagation de retenue.
En effet, on a :

$$X_{RNS} \ \Delta \ Y_{RNS} = (x_1 \ \Delta \ y_1 \bmod m_1, \ldots., x_m \ \Delta \ y_m \bmod m_m)$$

avec $\Delta \in \{+, -, \times\}$

Les multiplications et les additions/soustractions sont donc découpées en petites opérations indépendantes de w bits. Ces opérations sont parallèles de façon naturelle, permettant d'obtenir très rapidement le résultat d'un produit ou d'une somme. La figure III.5 illustre le parallélisme du RNS et l'indépendance des calculs sur les différents canaux.

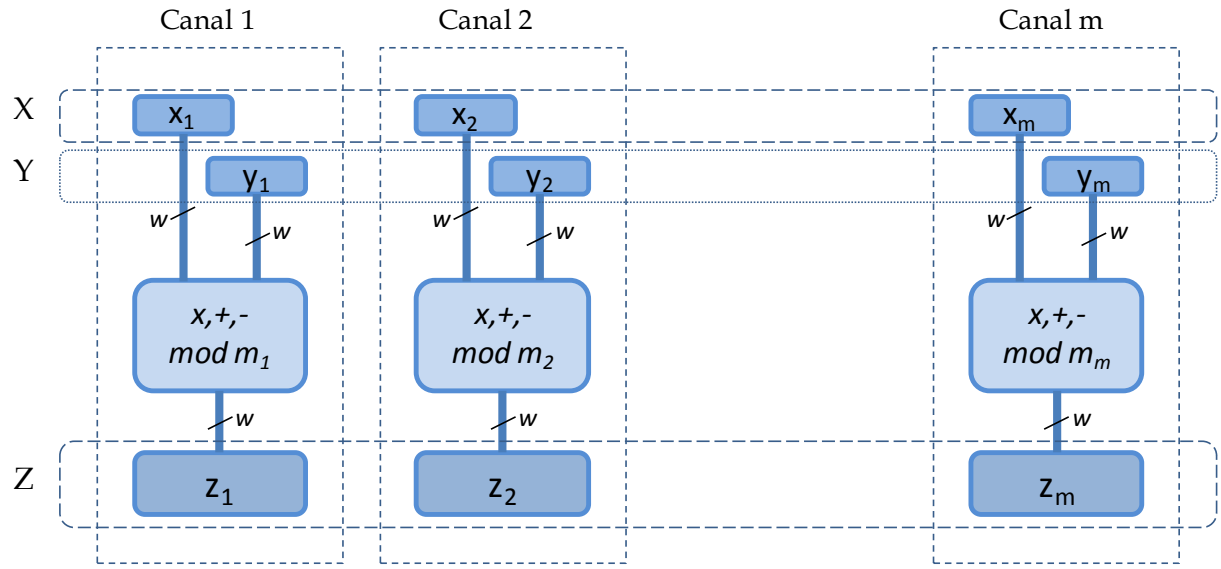

Figure III.5. Calcul dans le domaine RNS [Big14]

En appliquant le RNS aux coefficients $(a_i)_{i=0..n}$ et $(b_j)_{j=0..n}$ de taille $log_2 q$ =*Coeff_L*, nous les décomposons en des petits coefficients de taille *Coeff_S*. Le produit des coefficients $(a_i . b_j)$ décrit dans l'algorithme de multiplication classique $A(x) \times B(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j \, x^{i+j}$ sera alors transposé dans le domaine RNS.
Notons que la décomposition des coefficients ou représentation RNS des coefficients sera faite une seule fois au début des calculs et qu'une opération inverse IRNS est nécessaire à la suite de chaque produit. La réduction modulaire entière est également effectuée dans le domaine RNS.
Pour le choix des paramètres relatifs à cette représentation RNS (Base RNS, w et l), nous nous référons aux études effectuées dans ce domaine comme par exemple [BM04] et [Big14].

III.4.3. Partitionnement logiciel/matériel

En raison de la taille des opérandes que nous manipulons et de la complexité des calculs, implanter un multiplieur polynomial modulaire avec les caractéristiques que nous avons cité purement en matériel en ciblant des FPGAs n'est pas adapté. Une solution consiste à proposer un partitionnement logiciel/matériel pour nos calculs. Nous aurons ainsi un processeur généraliste communiquant avec un accélérateur matériel via un ou des bus de données. Ce processeur peut être un processeur embarqué (ARM par exemple) communiquant avec la partie programmable d'un SOPC ou un ordinateur communiquant avec une carte FPGA.
Côté logiciel, le processeur exécute les calculs simples et moins coûteux en temps afin d'éviter toute latence pénalisante au niveau des performances. Ces calculs concernent la décomposition en sous-polynômes et la génération de la base RNS en début des traitements, et la reconstruction du polynôme résultat et la réduction polynomiale modulaire, effectuées une seule fois à la fin des calculs. La partie matérielle implante principalement la multiplication polynomiale modulaire avec les transformations vers et du domaine RNS et le produit des coefficients (Figure III.6).

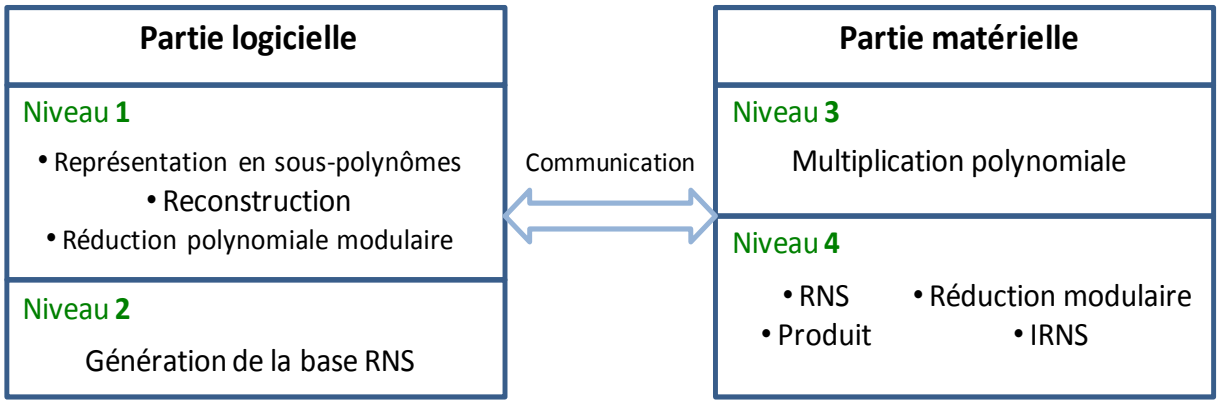

Figure III.6. Partitionnement logiciel/matériel

L'évaluation des performances de cette architecture en terme de durées de calcul doit tenir compte de trois éléments :

- la latence des calculs exécutés en matériel
- la latence des calculs exécutés en logiciel
- le coût du transfert des données : avec notre approche, les données à échanger avec la partie matérielle sont les modules de la base RNS, les coefficients des polynômes à multiplier, le module q et les coefficients du polynôme résultat.

Pour tirer profit de la flexibilité de cette architecture et obtenir un bon facteur d'accélération, il faut que la somme des latences du calcul logiciel et des transferts soit négligeable devant les calculs exécutés par la partie matérielle.

Cette section présente l'approche d'un point de vue global. Nous avons tenté de répondre aux exigences que nous nous sommes fixé en proposant un multiplieur polynomial générique et flexible, capable de supporter des polynômes très grands, et pouvant être implanté sur des

cibles matérielles de caractéristiques variées. La section III.6 donnera davantage de détails sur les implantations et les résultats.

III.5. Synthèse de haut niveau appliquée à la cryptographie homomorphe

Le flot de conception d'un circuit constitue l'ensemble des étapes et outils dont un développeur d'applications a besoin afin d'obtenir un circuit spécifique (ASIC) ou un FPGA mettant en œuvre l'application souhaitée. On distingue deux approches : la première dite classique (figure III.7) et la deuxième qui exploite une synthèse dite de haut niveau (figure III.8). Nous nous focaliserons par la suite sur des implantations à base de FPGA. Toutefois, l'approche que nous proposons peut tout aussi bien s'appliquer pour concevoir un ASIC.

Dans une approche classique, le concepteur décrit son architecture dans un langage de description matérielle (HDL - Hardware Description Langage) tel que VHDL, Verilog ou encore System Verilog. Une opération de synthèse logique, avec un logiciel approprié, décompose cette description selon les primitives matérielles réelles du FPGA (LUT, FF, etc). Le niveau de description obtenu est appelé netlist. Cette opération fournit également une première estimation des caractéristiques matérielles précises du circuit (classiquement, en surface et fréquence). Le concepteur évalue la conformité par rapport à ses contraintes et objectifs et, au besoin, modifie la description HDL afin d'obtenir une meilleure solution. La dernière étape, le placement- routage, vise à sélectionner les composants du FPGA qui mettent en œuvre le circuit et à les connecter. Enfin, une fois le flux binaire de configuration (bitstream) obtenu, il ne reste plus qu'à programmer le FPGA. Étant donné l'utilisation d'un langage de description matérielle comme entrée à ce flot, le développeur doit posséder de solides compétences en matériel afin de générer des circuits performants. De plus, toute nouvelle configuration au niveau des paramètres définissant l'application demande des efforts conséquents et peut être facilement sujette à des erreurs.

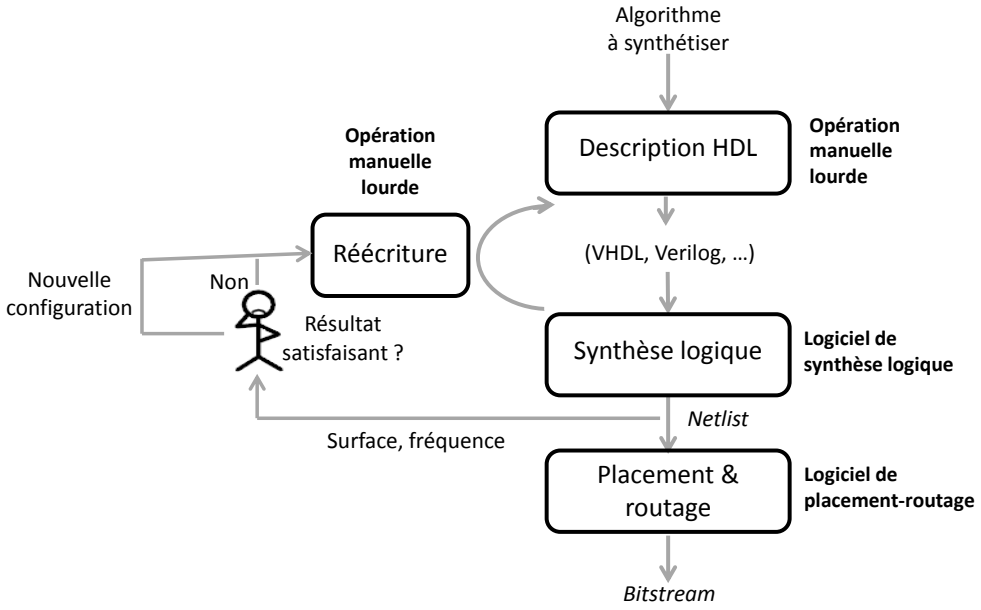

Figure III.7. Flot de conception classique

Afin de réduire le temps de développement d'un circuit et d'éviter des étapes de traduction algorithme/description matérielle, une autre couche d'abstraction peut être introduite. Le point d'entrée d'un flot utilisant la synthèse de haut niveau (HLS pour High-Level Synthesis) n'est plus une description dans un langage HDL mais plutôt une description dans un langage dit de haut niveau (C, C++, etc). La description de la fonctionnalité du circuit à concevoir est donnée sous forme d'un algorithme, ce qui facilite le développement et offre un gain en temps considérable. À la sortie de l'étape de synthèse de haut niveau, le développeur obtient un ou plusieurs fichiers décrivant son application dans un langage HDL. Dans ce flot de HLS, l'utilisateur peut donner des consignes spécifiques sur la structure du circuit à construire (type des opérateurs et leur nombre, degré de partage des opérateurs, dimensionnement des mémoires, etc). Après génération, les caractéristiques matérielles du circuit final (principalement surface, fréquence et latence en cycles d'horloges) sont exposées à l'utilisateur pour analyse. Celui-ci peut alors choisir de conserver la solution obtenue, ou bien de tenter d'en trouver une meilleure en agissant sur des commandes spécifiques du logiciel de HLS.

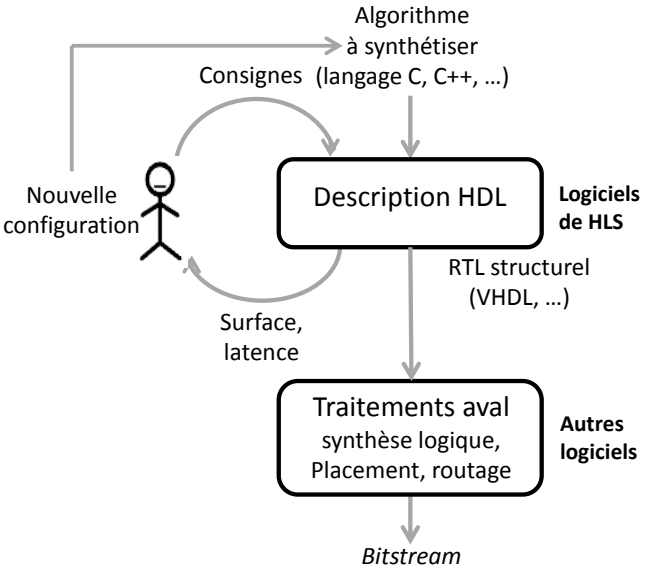

Figure III.8. Flot de conception avec synthèse de haut niveau

C'est un flot de type HLS qui a été choisi pour les travaux présentés dans ce manuscrit. En effet, visant un multiplieur polynomial modulaire générique et flexible capable de s'adapter facilement et rapidement à l'évolution remarquable du chiffrement homomorphe et aux nouveaux dimensionnements que connaissent perpétuellement les schémas en fonction des contraintes des applications (sécurité, efficacité, etc), ce type de flot se révèle une alternative intéressante au flot classique. Il permet au concepteur d'avoir de nouvelles solutions répondant à de nouvelles configurations (degré, taille des coefficients) et/ou de nouvelles contraintes (type de cible matérielle, ressources, latence, etc) en un temps et avec un effort réduits.

III.6. Implantations et résultats

III.6.1. Aspects pratiques

Nous allons commencer par spécifier les outils dont nous allons nous servir et les étapes que nous allons suivre lors des implantations.

Outil de synthèse de haut niveau :
Le logiciel de HLS choisi pour générer les différents circuits est l'outil AUGH développé par Prost-Boucle et al. [PB+14], gratuit et sous licence libre. Ce logiciel a été spécifiquement conçu pour générer automatiquement des accélérateurs matériels sous contrainte de ressources. AUGH accepte en entrée un sous-ensemble du C ANSI et produit des circuits en VHDL. Il supporte un grand nombre de FPGAs (Xilinx, Altera, Lattice).
L'utilisateur n'est pas investi dans les transformations du circuit en vue de sa mise en œuvre matérielle. C'est le logiciel AUGH, à qui l'utilisateur fournit une contrainte en ressources et une contrainte en fréquence, qui décide des transformations à appliquer au circuit. Il est aussi possible de piloter l'outil pour des étapes de conception qui ne relèveraient pas d'un algorithme : définition du niveau hiérarchique supérieur du circuit, types d'entrées/sorties et interfaces de communication, paramétrage de certains algorithmes d'élaboration et transformation, etc.
Outre son autonomie, AUGH est un outil de HLS rapide car les itérations appliquées en vue d'obtenir une description fonctionnelle sont logicielles et l'on dispose en quelques secondes d'estimateurs sur le circuit matériel en amont des outils du vendeur de FPGA. Ainsi, il permet une grande facilité d'exploration, guidant le concepteur dans son processus d'amélioration.

Autres outils :
Toutes les simulations et tests sont réalisés avec le simulateur Modelsim de Mentor Graphics.
Toutes les implantations sur cibles Virtex 7 ou Spartan 6 ont été obtenues avec la suite "ISE 14.6" de Xilinx, tandis que celles sur Zybo sont générées avec la suite "Vivado 15.2".

Etapes mettant en œuvre le flot proposé :
La figure III.9 décrit les outils que nous avons utilisé pour mettre en œuvre l'approche proposée. Le langage de haut niveau que nous utilisons est le langage C et toute nouvelle configuration de paramètres est réalisée à ce niveau. Lors de la configuration, on spécifie le degré n, le module q, le degré du bloc de base K et la base RNS.
La vérification fonctionnelle est faite par comparaison avec des vecteurs de référence générés avec Sage. Le logiciel AUGH effectue une synthèse de haut niveau et génère des estimateurs et une description matérielle au niveau transfert de registres (RTL) en VHDL à partir de la description de haut niveau. Une fois la description matérielle générée et les estimations données par AUGH approuvées, la synthèse et le placement-routage sont réalisés.

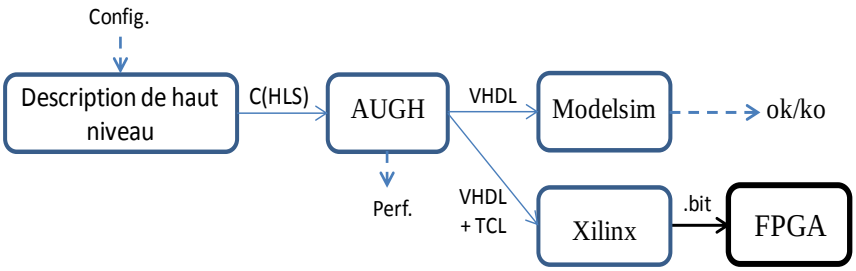

Figure III.9. Illustration du flot proposé

III.6.2. Evaluation des performances

Nous allons commencer cette partie par une évaluation des circuits générés par l'outil de HLS en les comparant avec des circuits de l'état de l'art conçus manuellement. Dans une seconde partie, nous illustrerons la flexibilité de notre approche et montrerons comment nous avons répondu aux quatre critères fixés dans la section III.4.1. Enfin, nous conclurons par un exemple complet de multiplieur polynomial modulaire de type co-conception.

Rappelons les paramètres permettant de configurer notre architecture :

(1) Le degré et la taille des polynômes à multiplier (n, $\log_2 q$);
(2) La taille K du bloc *Polynome_K ;*
(3) La base RNS ;
(4) Le polynôme réducteur f(x).

III.6.2.1. Implantations sur Spartan et Virtex

Afin d'évaluer les performances des différents circuits générés, nous allons commencer par une comparaison avec les travaux de l'état de l'art qui ont proposé une implantation purement matérielle du multiplieur polynomial. Pour ce faire, nous spécifions les mêmes paramètres que ceux présentés dans les différents travaux et nous ciblons des FPGAs de la même famille. La taille des coefficients ne dépassant pas 32 bits, nous n'avons pas décomposé les coefficients et nous avons décomposé les polynômes seulement dans le cas où n=32768. Pour les autres valeurs, nous instancions les valeurs de n et q, générons la description matérielle avec AUGH et passons par le reste du flot. Les résultats sont résumés dans le tableau III.4.

Le tableau III.4 montre que notre approche, bien que générant des circuits moins rapides que les conceptions optimisées manuellement, consomme beaucoup moins de ressources que les deux circuits présentés par exemple dans [DNF+15] ; nous obtenons une réduction d'un facteur 9 en moyenne. Cette différence significative s'explique par le choix de l'algorithme de multiplication polynomiale, à savoir la NTT dans [DNF+15]. Ce dernier conduit à une latence inférieure à celle de notre accélérateur grâce à sa faible complexité algorithmique, mais présente plusieurs contraintes et nécessite une grande quantité de ressources matérielles.

Tableau III.4. Comparaison avec les travaux de l'état de l'art

Références	Configuration			Ressources				
	n	$\log_2 q$	Plateforme	Slice LUT	Slice Register	DSP	Bram	Latence (μs)
[DNF+15] (1)	1024	26	Spartan 6	10801	3176	0	0	40,98
[DNF+15] (2)	1024	26	Spartan 6	2464	915	16	14	32,28
Mes travaux	1024	26	Spartan 6	182	114	3	10	69,1
[JCM+15]*	512	32	XC7VX1140T	252341	130826	512	2048	4,11
Mes travaux (3)	512	32	XC7V585T	7032	920	368	0	5,27
Mes travaux (4)	512	32	XC7V585T	171	102	3	3	66,41
[DOS+15]	32768	32	XC7VX690T	219192	90789	139	768	$9{,}51 \times 10^3$
Mes travaux (5)	32768	32	XC7V585T	3392	1920	48	792	$41{,}12 \times 10^3$
Mes travaux (6)	32768	32	XC7V585T	13568	7680	192	792	$10{,}28 \times 10^3$

(1) Données stockées dans des Lutrams (2) Données stockées dans des Brams (3) LUTs (4) DSPs et Brams
(5) Notre approche sans optimisations (6) Parallélisation * Résultats de synthèse

Pour comparer avec [JCM+15], où les auteurs mettent en œuvre l'algorithme classique, nous présentons deux solutions dans le tableau III.4. Afin d'exploiter le parallélisme et d'obtenir une latence minimale, nous appliquons une directive d'optimisation à AUGH qui interdit le partage d'opérateurs. Dans ce cas, seuls les LUTs sont utilisés pour stocker les coefficients et un nombre maximal de DSP peut être facilement parallélisé (cas 3). Dans l'autre conception (cas 4), les BRAMs sont utilisées pour stocker des informations, ce qui limite le nombre de DSPs utilisables efficacement. Il faut toutefois souligner que nos résultats sont après placement - routage, tandis que [JCM+15] donne uniquement des résultats après synthèse.

Pour les polynômes ayant des degrés de 32768, nous décomposons les polynômes opérandes en des sous-polynômes de degré 8192 et nous commençons par appliquer notre approche sans aucune optimisation. Dans ce cas (cas 5), les ressources matérielles requises par notre conception sont réduites d'un facteur 28 en moyenne. En contrepartie, notre conception a une latence de 41 ms, tandis que les auteurs de [DOS+15] indiquent une latence d'environ 9,5 ms. Étant donné que notre accélérateur possède une consommation relativement faible en ressources matérielles et que nous avons une dépendance très faible de données, nous pouvons lancer les calculs sur 4 instances du bloc *Polynome_K* en parallèle. Avec une telle optimisation, le circuit (cas 6) a une latence comparable à [DOS+15], tout en conservant un grand avantage au niveau des ressources utilisées.

Globalement, nos circuits générés nécessitent beaucoup moins de ressources que ceux de l'état de l'art et la généricité ainsi que la rapidité de conception sont favorisées par le flot employé. Côté performances, ces circuits sont compétitifs en comparaison avec des conceptions faites à la main.

Grâce à la flexibilité de notre approche, nous pouvons à partir de la même description générique de haut niveau produire plusieurs circuits avec différents compromis surface/latence et laissons à l'utilisateur la liberté de choisir le circuit répondant au mieux à ses besoins.

III.6.2.2. Illustration de la flexibilité de notre approche

La flexibilité a été déterminante dans notre architecture. Elle est mise en œuvre via les paramètres variables de notre approche. Une fois (n, $\log_2 q$) fixés, nous passons à la décomposition des polynômes opérandes au niveau du degré et des coefficients.

Côté coefficients, nous nous contentons de choisir une base RNS en s'appuyant sur les études réalisées dans ce domaine car cet aspect a été largement abordé dans la littérature et sort un peu du cadre de cette thèse. Nous avons choisi un ensemble de configurations de (n, $\log_2 q$) extraits de [LN14]. Ces configurations, reportées dans le tableau III.5, concernent des polynômes avec des degrés allant de 1024 à 16384 pour des coefficients allant de 19 bits à 1550. Cela permet d'atteindre une profondeur multiplicative entre L=0 et L=50.
La figure III.10 montre les performances des circuits générés pour chaque configuration. Les ressources représentent une moyenne des (LUT, registre, DSP et Bram) nécessaires. Les décompositions des coefficients et des polynômes opérandes sont appliquées lorsque nous traitons des grandes tailles et des grands degrés respectivement.

Tableau III.5. Exemple de paramètres (n, $\log_2 q$) pour $\lambda = 80$ *bits* et $\omega = 2^{32}$ pour différentes profondeurs multiplicatives L

	Yashe		**FV**	
L	n	$log_2(q)$	n	$log_2(q)$
0	1024	20	1024	19
1	2048	64	2048	43
10	4096	306	4096	271
10	8192	326	8192	292
50	16384	1550	16384	1473

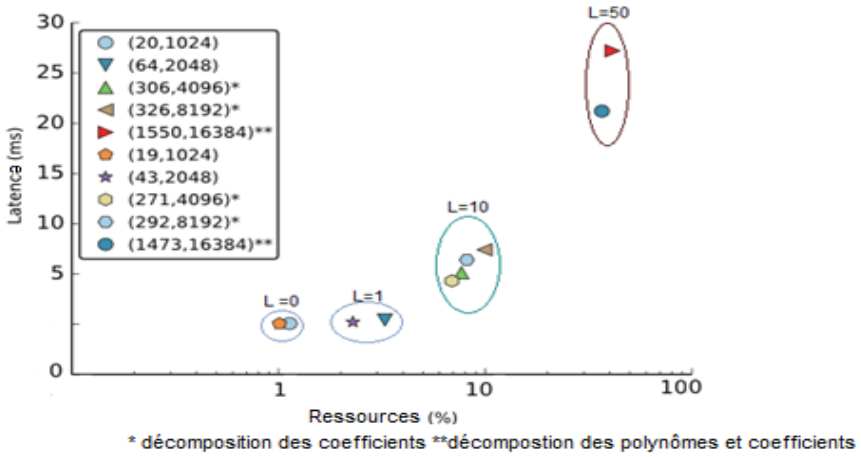

Figure III.10. Performance des circuits pour différentes configurations de ($\log_2 q$, n) correspondant à différentes profondeurs multiplicatives L sur une cible de type Virtex 7

Côté degré, nous nous intéressons à la valeur de K, taille du bloc *Polynome_K*. Le choix de la valeur de K est un point important car il va influencer une bonne partie du déroulement des calculs. En effet, K détermine la taille du multiplieur polynomial *Polynome_K* qui va être implanté sous forme matérielle, le nombre de réitérations du bloc *Polynome_K* et la quantité des données qui vont être émises vers et de l'accélérateur au fur et à mesure des calculs.
IL faut noter qu'il n'existe aucune restriction particulière sur ce choix à part les ressources disponibles sur la cible matérielle. Par contre, si nous voulons optimiser les performances de nos circuits, il faut choisir K de manière à :

- avoir une répartition équilibrée des coefficients des polynômes opérandes entre les différents blocs *Polynome_K ;*
- permettre une parallélisation des calculs sur les différents blocs *Polynome_K* compte tenu de la dépendance très faible de données ;
- assurer une synchronisation entre l'envoi et la réception des données et le déroulement des calculs.

Nous avons fait un travail d'exploration afin de déterminer la meilleure décomposition permettant de respecter un compromis précis entre l'utilisation efficace des ressources matérielles et l'obtention de durées de calculs compétitives. Ce travail concerne trois configurations de (n, $\log_2 q$) à savoir (8192, 64), (16384, 64) et (32768, 64). Pour chacune d'entre elles, nous avons fait varier K. La figure III.11 montre les deux solutions retenues pour chaque couple (n, $\log_2 q$). Les ressources représentent une moyenne des (LUT , registre, DSP et Bram) nécessaires pour avoir le résultat final. La solution est optimisée quand nous employons plusieurs instances du bloc *Polynome_K* en parallèle.

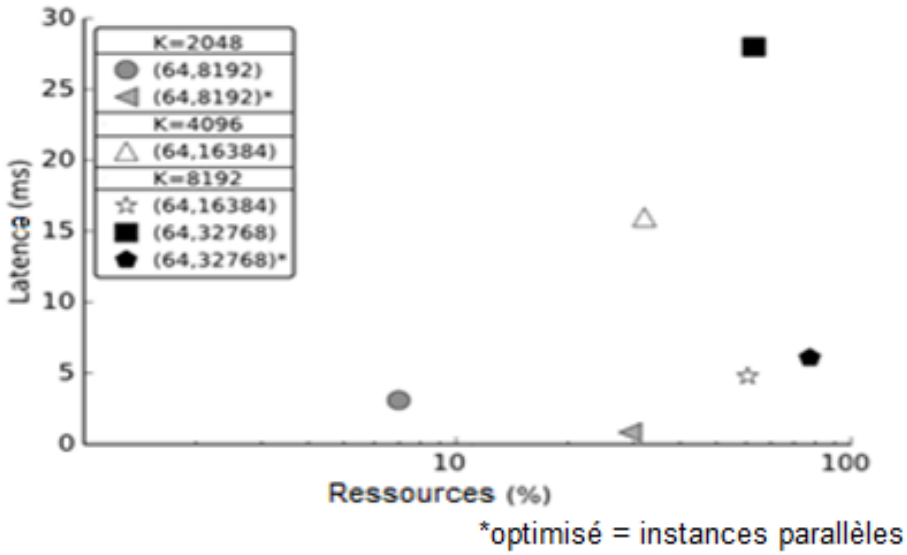

Figure III.11. Performance du multiplieur pour des coefficients de taille 64 bits et différents degrés, avec plusieurs décompositions sur une cible de type Virtex 7

Outre sa flexibilité, notre multiplieur a aussi la particularité d'être capable de supporter des opérandes de grande taille et d'être évolutif (pouvant s'adapter aux ressources disponibles sur la cible matérielle). Si nous avons besoin par exemple de polynômes de degré n=80000 ayant

des coefficients de taille 32 bits, les ressources disponibles sur un FPGA de type Virtex 7 (XC7V585T) ne suffisent pas pour calculer la multiplication polynomiale et stocker les opérandes d'entrée et de sortie. AUGH peut nous donner une estimation des ressources nécessaires si nous définissons une contrainte en surface supérieure à 100%. Le tableau III.6 donne une estimation des ressources requises si nous définissons n = 80000 et les résultats si nous considérons notre approche avec une décomposition des polynômes en sous polynômes de degré K=4000. Dans ce cas, le bloc de base *Polynome_K* sera appelé 20^2 fois afin d'obtenir le résultat final.

Tableau III.6. Illustration du caractère évolutif de notre approche

	Slice Register (%)	**LUT** (%)	**Bram** (%)	**DSP** (%)
Estimations du logiciel AUGH sans décomposition	--	140	150	12
Décomposition (4000, 20^2)	7	25	90	95

En conclusion, nous résumons dans la figure III.12 les paramètres (n, $\log_2 q$) couverts dans cette section en comparaison avec les travaux [DNF+15], [JCM+15] et [DOS+15] de l'état de l'art. Pour chaque configuration de (n, $\log_2 q$), nous pouvons générer plusieurs circuits respectant différents compromis ressources/latence. Le développement d'un nouveau circuit prend environ 3 heures en moyenne, ce qui est très petit en comparaison avec un nouveau circuit fait "à la main". Les résultats confirment la flexibilité de notre approche et montrent qu'elle offre des résultats pertinents dans de nombreux domaines d'application. Les résultats présentés ici pourraient être facilement complétés pour d'autres tailles de coefficients et d'autres degrés.

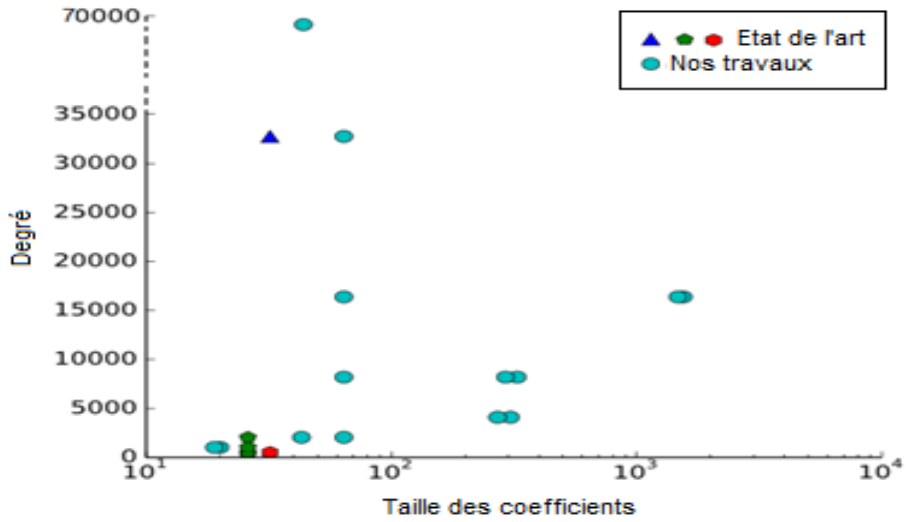

Figure III.12. Différentes configurations (degré, taille des coefficients) implantées dans ce travail en comparaison avec l'état de l'art

III.6.2.3. Implantation sur une plateforme Zynq

Cette section est consacrée à l'évaluation du multiplieur polynomial modulaire complet où les calculs sont partagés entre la partie logicielle et la partie matérielle. Rappelons que les calculs incluent la multiplication des polynômes, la réduction entière des coefficients et la réduction polynomiale modulaire.
Dans ce cas, la comparaison de notre accélérateur avec l'état de l'art est délicate. Cela peut s'expliquer par les raisons suivantes :

- Nous avons montré que notre approche permet de couvrir les mêmes jeux de paramètres que l'état de l'art au niveau de (n, $\log_2 q$). Il reste à configurer f(x). Le choix populaire f(x)=$x^n + 1$, fixé dans la plupart des travaux, interdit l'utilisation du batching. En plus, ces travaux utilisent la NWC qui permet de réaliser la réduction modulaire polynomiale pendant la FFT. Ainsi, les comparer avec notre travail qui favorise une forme générale de f(x) et traite la réduction polynomiale modulaire séparément ne serait pas correct.
- Les travaux favorisant la forme générale de f(x) utilisent une technologie matérielle différente (Xilinx vs Altera).
- Pour avoir une comparaison fiable, il faudrait également se comparer à une solution de type co-conception qui prend en compte à la fois la latence des transferts et la réduction polynomiale modulaire.

Trouver des implantations parfaitement comparables avec notre travail au niveau des points évoqués n'a pas été possible. Par conséquent, nous avons décidé de comparer nos circuits avec des implantations logicielles et ce en termes de temps de calcul uniquement. En termes de ressources, nous avons déjà montré que notre approche utilise une quantité de mémoire et de ressources bien inférieures aux implantations basées sur la FFT ou Karatsuba. Cette conclusion reste valable pour la suite.

Pour évaluer les performances globales de notre prototype, nous avons choisi la plateforme Zybo Zynq-7000 dotée d'un processeur ARM Cortex-A9 double cœur et d'une partie programmable (PL) dont la logique est équivalente à un FPGA Artix-7. Le but étant de montrer le facteur d'accélération dans le cas d'une architecture logicielle/matérielle, nous n'avons pas choisi de grands degrés et tailles des coefficients pour les polynômes opérandes. Les ressources disponibles au niveau de la Zybo sont alors suffisantes. Néanmoins, l'évaluation resterait la même pour une carte avec un FPGA plus grand, telle que la ZC702 par exemple. Notre réalisation repose sur un cœur ARM et un accélérateur matériel sollicité durant l'opération de multiplication polynomiale. La communication entre le processeur et la partie PL est assurée avec les bus système de type AXI. Nous utilisons les bus "high performance axi bus" (HP0–HP3) pour envoyer les coefficients des polynômes opérandes et recevoir les coefficients des polynômes résultats et les bus "general purpose axi bus" (GP0–GP3) pour le transfert des données d'initialisation.

Pour l'étude des résultats d'implantation, nous avons fixé n=2048 et $log_2 q = 43$ bits. Ce paramétrage nous permet d'atteindre une profondeur multiplicative de 2 avec le schéma FV pour $\lambda = 80\ bits$ et $\omega = 2^{32}$. Nous avons généré un polynôme irréductible f(x) de degré 2048.

Nous avons décomposé les polynômes opérandes en 4 sous polynômes de degré 512 chacun. Ainsi, on obtient :

$$A \equiv \{A_1, A_2, A_3, A_4\} \text{ et } B \equiv \{B_1, B_2, B_3, B_4\}$$

Nous avons déjà montré que les circuits générés par l'outil de HLS nécessitent une quantité réduite de ressources et que nous avons une dépendance très faible de données suite à la décomposition des polynômes. Par suite, nous pouvons lancer les calculs en parallèle sur plusieurs instances du bloc *Polynome_K.* On commence par envoyer les deux premiers sous-polynômes A_1 et B_1 et la multiplication polynomiale est lancée. Les A_i et B_j sont ensuite envoyés au fur et à mesure, les polynômes résultats étant reçus pendant l'exécution des calculs sur les différentes briques du *Polynome_K* (figure III.13). Une fois tous les résultats obtenus, le polynôme produit final est reconstruit. Cette étape est réalisée en logiciel et correspond à des ré-ordonnancements et additions des coefficients. Finalement, le polynôme produit final est réduit modulo f(x).

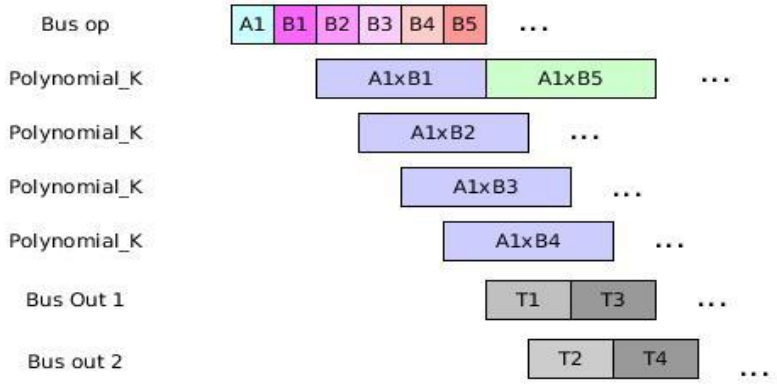

Figure III.13. Parallélisation des calculs sur plusieurs instances du bloc de base *Polynome_K*

En appliquant la parallélisation sur 4 instances du bloc *Polynome_K,* on gagne un facteur d'accélération d'environ 3 par rapport à une solution sans parallélisme.

Avec le paramétrage mentionné, nous obtenons le résultat final $R(x) = A(x) \times B(x) mod\ (p, f(x))$ en 2,6 ms. La même opération implantée avec Sage demande 21,4 ms sur un Intel Core i5-2450M (2,5 GHz).

En analysant ces résultats, nous avons remarqué que :

- la latence des transferts est négligeable devant la latence des calculs s'exécutant en matériel ;
- la latence de la réduction modulaire polynomiale est la plus importante. Ceci s'explique par la forme générale du polynôme réducteur (un nombre de coefficients non nuls très faible). Fixer f(x) à $x^n + 1$ (avec seulement deux coefficients non nuls), par exemple, diminue significativement cette latence. Ainsi, une étude détaillée sur le choix du polynôme réducteur et une amélioration de l'algorithme de réduction modulaire polynomiale implanté sont des pistes à explorer.

III.7. Conclusion

Dans ce chapitre, une nouvelle façon d'aborder la multiplication polynomiale modulaire a été proposée pour les schémas homomorphes à base du problème RLWE. La caractéristique principale est de pouvoir supporter des grands polynômes avec des degrés et tailles de coefficients arbitraires. Nous avons proposé un algorithme basé sur la décomposition des opérandes aussi bien au niveau des polynômes (degré) que des coefficients (à l'aide de la représentation RNS). Notre approche basée sur la synthèse de haut niveau permet d'obtenir rapidement des circuits compétitifs. Elle peut facilement s'adapter aux contraintes spécifiées par l'utilisateur. Des optimisations basées essentiellement sur la parallélisation des calculs ont été proposées et ont permis de réduire la durée globale des calculs tout en maintenant un gain sensible en ressources par rapport à des implantations optimisées manuellement et beaucoup moins adaptables.

Le multiplieur polynomial présenté va être utilisé dans le chapitre IV pour construire des primitives du schéma FV.

Chapitre IV. Extension de l'approche proposée et application dans un contexte de chiffrement homomorphe

Dans ce chapitre, nous allons nous intéresser à une deuxième fonction nécessaire dans la construction des schémas homomorphes à base du problème RLWE, à savoir l'échantillonnage gaussien discret. Les résultats du profilage ont montré que cette fonction est aussi coûteuse. Ainsi, une accélération matérielle de cette fonction aura un impact sur les performances du cryptosystème. Nous allons adopter l'approche proposée dans le chapitre III et utiliser la synthèse de haut niveau pour la génération de la fonction d'échantillonnage. Une description de cette fonction sera présentée dans la section IV.1. La section IV.2 détaillera l'implantation et les résultats. Nous utilisons dans la section IV.3 le multiplieur polynomial modulaire et le circuit d'échantillonnage pour construire des primitives du schéma FV et concluons ce chapitre par une discussion sur notre approche.

IV.1. Echantillonnage Gaussien Discret

IV.1.1. Présentation

Les distributions gaussiennes continues sont connues depuis longtemps. Ils sont des objets souvent utilisés en théorie des probabilités, en particulier à cause du théorème de la limite centrale (la moyenne d'un grand nombre de valeurs aléatoires suit une distribution proche d'une gaussienne, quelle que soit la distribution de départ) mais aussi pour leurs propriétés géométriques. Leurs versions discrètes introduites par Gentry, Peikert et Vaikuntanathan [GPV08] sont l'un des éléments de base fondamentaux de la cryptographie à base de réseaux. Depuis, l'échantillonnage gaussien a permis de réaliser des fonctionnalités avancées à savoir des IBE (schémas de chiffrement basé sur l'identité) hiérarchiques [CHK+10, ABB10], des modèles de signature standards [ABB10, Boy10], du chiffrement par attributs [BGG+14] et des schémas homomorphes [BV11a, BV11b, FV12, BLLN13].

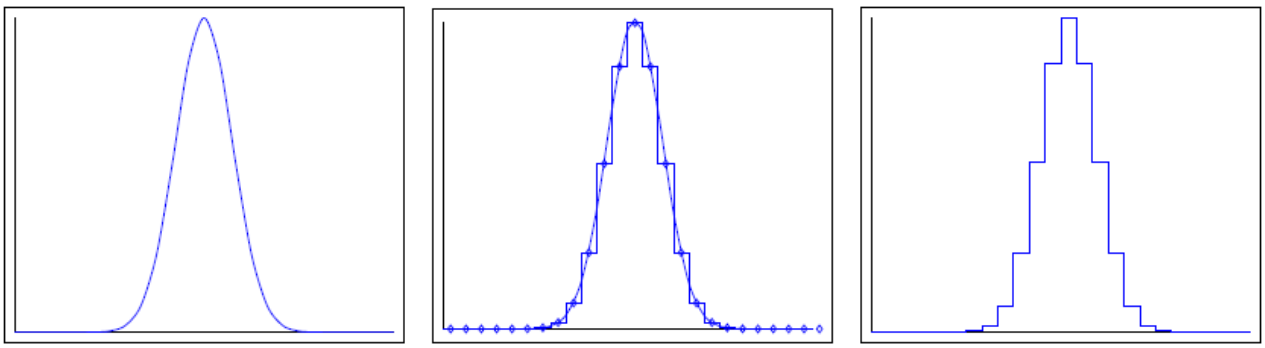

Figure IV.1. Exemple de distribution Gaussienne continue et discrète

Définition :
Soit $L \subset \mathbb{R}^n$ un réseau de rang plein. La distribution gaussienne (ou normale) discrète notée $D_{L,c,\sigma}$ de support L, de moyenne c et d'écart type σ est définie par :

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

Avec $\rho_{c,\sigma}(x) = \exp(-\|x - c\|^2 / 2\sigma^2)$.
On trouve également $\rho_{c,\sigma}(x) = \exp(-\pi\|x - c\|^2 / s^2)$ avec $s = 1 + 2\sum_{z=1}^{\infty} e^{-z^2/2\sigma^2} \simeq \sigma\sqrt{2\pi}$
Les indices L et c sont omis lorsque $L = \mathbb{Z}^n$ et c=0.

Dans ce chapitre, nous utiliserons des variantes discrètes de la loi gaussienne, dont le support L est un réseau euclidien. La distribution est la même que celle d'une gaussienne continue mais tous les points échantillonnés appartiennent au support L.
Les propriétés des gaussiennes discrètes sont très souvent semblables à celles des gaussiennes continues. En particulier, la queue de la distribution a un poids très faible. D'autres propriétés des gaussiennes continues s'adaptent aux gaussiennes discrètes quand le paramètre s est suffisamment grand. Ceci est quantifié à l'aide du paramètre de lissage du réseau L et correspond à la plus petite quantité de bruit gaussien à déposer sur chaque point de L pour le faire "disparaître" [MR07].

Echantillonnage gaussien discret unidimensionnel (one-dimensionnal gaussian sampling) :
L'échantillonnage gaussien utilisé en cryptographie homomorphe est l'échantillonnage gaussien unidimensionnel centré en 0 sur les entiers ($L = \mathbb{Z}$) (figure IV.2).

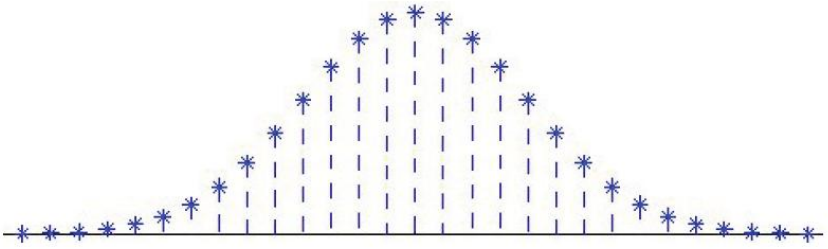

Figure IV.2. Exemple de Gaussienne discrète sur les entiers ($\boldsymbol{L} = \mathbb{Z}$)

Il convient de noter qu'échantillonner selon une distribution gaussienne discrète d'écart type σ sur $L = \mathbb{Z}^n$ $(n \geq 1)$peut être réduit à échantillonner n entiers selon une distribution gaussienne discrète d'écart type σ sur $L = \mathbb{Z}$ [DG14].

Définition :
Pour $L = \mathbb{Z}$, une distribution gaussienne discrète D_σ de moyenne c=0 et d'écart type σ est définie par :

$$D_{\mathbb{Z},\sigma}(E = z) = \frac{1}{s}\, e^{-(z)^2/2\sigma^2}$$

Avec : $s = 1 + 2\sum_{z=1}^{\infty} e^{-z^2/2\sigma^2} \simeq \sigma\sqrt{2\pi}$

IV.1.2. Paramètres pratiques

Les distributions gaussiennes ont naturellement des queues ("tails") très longues et des précisions très hautes. Afin d'évaluer les performances des échantillons, il est nécessaire de trouver des paramètres pratiques qui permettent de caractériser la distribution sans compromettre sa nature. Pour ce faire, on trouve dans la littérature [HKR+16] trois paramètres (σ, λ, τ) définis respectivement par :

- L'écart type σ : contrôle la forme de la distribution en mesurant la dispersion des données par rapport à la moyenne. La valeur de σ dépend des modules utilisés pour définir les schémas construits autour du (R)-LWE ou SIS *(Short Integer Solution)*.
- Le paramètre de précision (ou de sécurité) λ : contrôle le niveau de précision requis pour une implantation. Il exige que la distance statistique entre la distribution gaussienne discrète "parfaite" et la distribution pratique soit inférieure à $2^{-\lambda}$ (en relation directe avec le niveau de sécurité du schéma). Dans [Sa15], Saarinen montre que la précision doit dépasser $\lambda/2$ pour atteindre un niveau de sécurité λ.
- Le paramètre *tail-cut* τ : détermine combien de queues lourdes ("heavy tails") peuvent être exclues dans l'implantation pratique. Autrement dit, pour un niveau de sécurité de s bits, la distance entre la distribution cible et "parfaite" ne doit pas dépasser 2^{-s}. Ainsi, au lieu de considérer des échantillons $|x| \in \{0, +\infty\}$, il suffit d'avoir $|x| \in \{0, \sigma\tau\}$. Par suite, en appliquant cette condition à la précision, on obtient $\tau = \sqrt{\lambda \times 2\ln(2)}$.

IV.1.3. Méthodes d'échantillonnage gaussien discret

Il existe de nombreuses méthodes permettant de réaliser un échantillonnage selon une distribution gaussienne discrète. Le but n'étant pas de donner une liste exhaustive, nous présentons les méthodes les plus connues afin de déterminer l'algorithme à retenir pour une implantation matérielle en suivant un flot de synthèse de haut niveau.

IV.1.3.1. La méthode de rejet

La méthode de rejet ou d'acceptation-rejet [Neu51] se base sur le fait qu'une approximation, même grossière, de la distribution cible (une gaussienne dans notre cas) par une distribution de proposition (une loi uniforme par exemple) peut suffire à générer facilement des échantillons. En effet, la méthode de rejet permet d'échantillonner selon une distribution f étant donnée une source qui échantillonne selon une distribution g, en rejetant un échantillon x avec une probabilité égale à $\frac{f(x)}{M.g(x)}$ avec M un réel tel que :

$$M.g(x) \geq f(x) \text{ pour tout } x$$

La probabilité de ne pas rejeter un échantillon est alors $\frac{1}{M}$ et les valeurs de sortie sont distribuées selon f.

Bien que la méthode de rejet apparaisse simple, son exécution pratique est coûteuse en temps vu le nombre de rejets qui peut être important. En plus, pour chaque essai, de nombreux bits aléatoires sont nécessaires, ce qui est très coûteux dans le cadre d'une implantation matérielle.

IV.1.3.2. La méthode de l'inversion

La méthode de l'inversion s'appuie sur le résultat suivant :
Proposition: Soit F une fonction de répartition (ou fonction de distribution cumulative). Soit U une variable aléatoire uniforme sur [0,1].
Alors : $F^{-1}(U)$ est une variable aléatoire de fonction de répartition F.

Ainsi, pour générer des réalisations d'une variable aléatoire de loi F, il suffit de générer des réalisations d'une loi uniforme (de tirer aléatoirement des nombres entre 0 et 1) et de calculer l'image par la fonction réciproque de F de ces nombres.
En pratique, la méthode d'inversion utilise une table de distribution cumulative (CDT : cumulative distribution table) pré-calculée à partir de la fonction de répartition. On part du fait que pour une variable aléatoire uniforme dans [0,1], la variable aléatoire $D_\sigma^{-1}(U)$ suit la distribution recherchée sur $\mathbb{Z}$. On remplit la table CDT de la distribution gaussienne discrète ; les valeurs correspondent aux probabilités $p_z = \Pr[x \leq z : x \leftarrow D_\sigma]$ pour $z \in S$. Au moment de l'échantillonnage, on tire u selon une loi uniforme aléatoire définie sur [0,1] et on effectue une recherche binaire à travers la table CDT afin de déterminer $z \in S$ tel que $u \in [p_{z-1}, p_z)$. La valeur de sortie est alors z.

IV.1.3.3. La méthode de Bernoulli

Définition : Loi de Bernoulli
Une variable aléatoire discrète suit une loi de Bernoulli si :

$$P(X = 1) = p \text{ et } P(X = 0) = 1 - p \text{ avec } p \in [0,1]$$

Dans [DDL+13], Ducas et al. ont proposé un algorithme d'échantillonnage Gaussien discret basé sur la distribution de Bernoulli. L'avantage principal de cette technique réside dans le fait qu'elle réduit la probabilité de rejet et ce en commençant par échantillonner à partir d'une distribution intermédiaire (facile à échantillonner) puis de la distribution cible.

IV.1.3.4. L'algorithme de Knuth-Yao

Introduit dans [KY76], l'algorithme de Knuth et Yao utilise le modèle de marche aléatoire ("random walk model") afin de générer des échantillons à partir d'une distribution discrète non uniforme connue. Il introduit la notion d'arbre DDG ("Discrete Data Generator tree") pour représenter l'exécution d'un algorithme d'échantillonnage. L'arbre DDG est formé par deux types de nœuds : des nœuds intermédiaires et des nœuds terminaux. Un nœud terminal contient un point d'échantillonnage. Un nœud terminal génère deux nœuds fils au niveau suivant de l'arbre. Durant l'opération d'échantillonnage, une marche aléatoire est effectuée en partant de la racine de l'arbre DDG. Pour chaque passage d'un niveau au niveau suivant de l'arbre, un bit aléatoire est utilisé pour déterminer un nœud fils. L'opération d'échantillonnage se termine quand la marche aléatoire atteint un nœud terminal.

IV.2. Implantation et résultats

L'algorithme retenu en vue d'une implantation matérielle est l'algorithme d'inversion car :

- Il est simple à mettre en œuvre ;
- Il ne nécessite pas de grands pré-calculs ou des calculs arithmétiques complexes, à part la table CDT.

Le but est de générer des échantillons selon une Gaussienne discrète sur les entiers en utilisant le flot de synthèse de haut niveau présenté au chapitre III.
Nous avons généré un échantillonneur pour plusieurs couples (n, q). Les résultats sont résumés dans le tableau IV.1.

Tableau IV.1. Caractéristiques des échantillonneurs générés

n	q	Slice LUT	Slice Register	DSP	Bram
256	493	230	720	3	15
256	7681	280	792	9	19
512	12289	501	890	15	31

La comparaison de nos résultats avec des travaux de l'état de l'art n'a pas été possible. Cela peut s'expliquer par plusieurs raisons :

- Les travaux qui s'intéressent au côté pratique de l'échantillonnage gaussien discret à savoir des implantations logicielles et/ou matérielles ne sont pas liés au contexte de la cryptographie homomorphe ;
- A ce jour et dans un contexte de chiffrement homomorphe, seules des études théoriques existent ;
- Les jeux de paramètres sont différents vu les contextes différents.

Ainsi, afin d'évaluer les performances de nos circuits, nous les avons utilisés pour accélérer des primitives du schéma FV ; les résultats vont être présentés dans la section IV.3.

IV.3. Application : schéma FV

Dans cette section, nous présentons l'accélération de deux primitives du schéma FV, à savoir le chiffrement et la multiplication homomorphe.
Les accélérateurs incluront le multiplieur polynomial modulaire décrit dans le chapitre III. Le circuit de l'échantillonneur gaussien discret est employé avec le multiplieur pour accélérer la primitive du chiffrement. Grâce à la flexibilité de notre approche, les circuits du multiplieur polynomial modulaire et de l'échantillonneur gaussien sont facilement adaptables aux caractéristiques des primitives du schéma FV.

IV.3.1. Chiffrement du schéma FV

Rappelons l'expression de l'opération de chiffrement du schéma FV.

Rappel : Expression de la Multiplication homomorphe du schéma FV :

Chiffrement : $c \leftarrow \text{Enc}(pk, m)$

$pk = (b, a)$

On a $u \leftarrow \chi_{key}$ et $e_1, e_2 \leftarrow \chi_{err}$

$$\boldsymbol{c} = \left([\Delta m + bu + e_1]_q, [au + e_2]_q\right) \in R^2$$

L'opération de chiffrement du schéma FV fait appel à deux multiplications polynomiales $\{b.u\}$ et $\{a.u\}$. Il s'agit de multiplications polynomiales classiques que nous implantons en faisant appel au multiplieur polynomial décrit au chapitre III.

Les différents polynômes (vecteurs) u, e_1 et e_2 sont générés selon une distribution gaussienne discrète.

L'accélération de la primitive de chiffrement du schéma FV incluant les circuits du multiplieur polynomial et de l'échantillonneur s'exécute selon les étapes suivantes :

- génération de u, e_1 et e_2 en matériel en utilisant le circuit de l'échantillonneur
- multiplication polynomiale $\{b.u\}$ en utilisant le circuit du multiplieur
- multiplication polynomiale $\{a.u\}$ en utilisant le circuit du multiplieur
- exécution du reste des calculs en logiciel une fois les résultats obtenus de la partie matérielle

La comparaison de notre accélérateur de la primitive de chiffrement de FV avec l'état de l'art a été complexe car la majorité des travaux se focalise sur l'accélération de l'opération de multiplication polynomiale et son application à la multiplication homomorphe. De plus, comme nous l'avons vu, le chiffrement nécessite la génération de polynômes dont les coefficients sont distribués selon une gaussienne discrète dont la mise en œuvre est délicate.

Par conséquent, nous avons comparé nos résultats et plus précisément les durées de calcul de notre accélérateur avec les durées d'une implantation logicielle réalisée avec Sage sur un Intel Core i5-2450M (2,5 GHz).

Nous avons réalisé des comparaisons pour plusieurs paramètres dans deux cas différents :

- dans le premier cas, nous admettons que les polynômes u, e_1 et e_2 sont générés au préalable et nous utilisons le circuit du multiplieur pour construire le chiffrement.
  Nous avons obtenu un facteur d'accélération de 10 en moyenne par rapport à une implantation avec Sage ;
- dans le deuxième cas, nous avons utilisé à la fois le circuit du multiplieur et le circuit de l'échantillonneur. Le facteur d'accélération passe à une valeur moyenne de 4. Cette dégradation des performances peut s'expliquer par le transfert des données entre l'échantillonneur implanté en matériel et la partie logicielle qui exécute le reste des calculs (envoi et réception des échantillons).

IV.3.2. Multiplication homomorphe du schéma FV

Rappelons qu'une multiplication homomorphe revient à multiplier deux chiffrés à travers la couche de chiffrement.

Rappel : Expression de la Multiplication homomorphe du schéma FV :

$$\tilde{\boldsymbol{c}}_{mult} = (c_0, c_1, c_2) \text{ avec } \begin{cases} c_0 = [\lfloor \frac{t}{q} . c_{1,0} . c_{2,0} \rceil]_q \\ c_1 = [\lfloor \frac{t}{q} . (c_{1,0} . c_{2,1} + c_{1,1} . c_{2,0}) \rceil]_q \\ c_2 = [\lfloor \frac{t}{q} . c_{1,1} . c_{2,1} \rceil]_q \end{cases}$$

$$\boldsymbol{c}_{mult} = \text{ReLin } (\tilde{\boldsymbol{c}}_{mult} , evk)$$

Dans le cas du schéma FV, des points importants sont à noter :

- Un chiffré de FV étant formé par deux polynômes, la multiplication homomorphe correspond à une multiplication entre les polynômes des chiffrés ;
- Après la multiplication des polynômes, une opération de division et arrondi est nécessaire afin d'obtenir un chiffré avec la forme adéquate. Ainsi, il devient nécessaire de réaliser une multiplication polynomiale sans la réduction modulaire entière par q ;
- Le module q peut avoir une forme quelconque et atteindre des valeurs très grandes. Par conséquent, l'exécution de l'opération de division et arrondi aura un impact non négligeable sur les performances de la multiplication homomorphe ;
- L'opération de division et arrondi doit être effectuée après l'opération de multiplication polynomiale. Par suite, la réduction modulaire entière arrive en troisième position ;
- La multiplication homomorphe du schéma FV fait appel à quatre opérations de multiplication polynomiale.

Nous avons choisi la configuration (2048, 102) définissant respectivement le degré et la taille des coefficients et nous avons conçu l'architecture de la multiplication homomorphe du schéma FV en faisant appel à 4 instances de notre multiplieur polynomial. Ce dernier a été régénéré avec l'outil de synthèse de haut niveau AUGH en éliminant la réduction modulaire entière.
Le tableau IV.2 résume les durées de notre multiplieur en comparaison avec d'autres implantations faites à la main. Nous reportons juste les durées de calcul car les travaux [MRL+16b] et [PNP+15] utilisent une technologie différente.
Nous pouvons confirmer que nous obtenons des résultats compétitifs avec des solutions conçues avec le flot classique tout en maintenant un avantage en ressources, comme nous l'avons montré au chapitre III.

Tableau IV.2. Durée de la multiplication homomorphe du schéma FV

Références	n	$log_2\, q$	Durée (ms)
Nos travaux	2048	102	11,61
[MRL+16b]	2560	108	11,95
[PNP+15]	4096	125	15,46

IV.3.3. Accélération du schéma FV

Dans cette section, nous allons accélérer les primitives du schéma FV excepté la génération des clés en utilisant nos circuits de multiplieur et d'échantillonneur. Nous avons choisi la configuration de (127, 4096) pour la taille des coefficients et le degré.

Afin d'évaluer les performances du schéma FV, nous l'avons comparé à une implantation logicielle de Lepoint [Lep14] (Intel Core i7-2600, 3,4 GHz). Les résultats sont résumés dans le tableau IV.3.

Tableau IV.4. Comparaison des performances de notre implantation logicielle/matérielle du schéma FV avec une implantation logicielle de Lepoint (Intel Core i7-2600, 3,4 GHz)

	KeyGen (ms)	Enc (ms)	Dec (ms)	Add. Hom. (ms)	Mult. Hom. (ms)
Implantation logicielle	0,2	34	16	1,4	148
Nos travaux	--	31	2,1	0,3	17,21

Les résultats montrent un facteur d'accélération qui peut atteindre 8, ce qui justifie l'intérêt de l'approche proposé. La primitive de chiffrement n'est pas vraiment améliorée, comme le cas présenté dans la section IV.3.1. Ainsi, une étude plus détaillée de l'algorithme d'inversion et une exploration des autres algorithmes d'échantillonnage gaussien discret serait l'une des premières perspectives de ce travail.

IV.3.4. Optimisations

Étant donné que notre approche ne pose aucune contrainte sur le polynôme réducteur et qu'elle autorise par la suite le batching, nous allons nous intéresser dans cette section à l'évaluation des performances de nos circuits dans le cas où on applique cette optimisation.

Rappelons que le batching est une technique qui permet de regrouper plusieurs bits de messages dans un même chiffré. Elle propose d'utiliser un espace de textes clairs plus large en appliquant le théorème des restes chinois.

Théorème des restes chinois appliqué aux polynômes :
En partant du fait qu'un polynôme irréductible (cyclotomique) $\Phi_N(x)$ peut être factorisé (sous certaines conditions) en un produit $\prod f_i\,(x) mod\ k$, nous obtenons :

$$\frac{\mathbb{Z}_t[x]}{\Phi_N(x)} = \frac{\mathbb{Z}_t[x]}{\mathrm{f}_1(x)} \times \ldots\ldots \times \frac{\mathbb{Z}_t[x]}{\mathrm{f}_k(x)}$$

Ainsi, les calculs pourront se faire sur chacun "slot" (associé à $\frac{\mathbb{Z}_t[x]}{\mathrm{f}_i(x)}$) en parallèle.

Exemple:
Soient c et c' deux chiffrés auxquels on associe $(r_1(x), \ldots, r_k(x))\ et\ (r'_1(x), \ldots, r'_k(x))$ leurs représentations dans les "slots" respectifs.

Ainsi :
$c_{ADD} = Somme\ (c, c') \quad \rightsquigarrow \quad (r_1(x) + r'_1(x), \dots, r_k(x) + r'_k(x))$
Et : $c_{MULT} = Produit\ (c, c') \quad \rightsquigarrow \quad (r_1(x) \times r'_1(x)\ mod\ f_1(x), \dots, r_k(x) \times r'_k(x)\ mod\ f_k(x))$

Pour pouvoir appliquer le batching, il faut vérifier que notre polynôme cyclotomique, irréductible dans $\mathbb{Z}_q[x]$, est réductible dans l'espace des messages, c'est-à-dire dans $\mathbb{Z}_t[x] = \mathbb{Z}_2[x]$ (dans notre cas). Ce qui exclut le choix $x^n + 1$.

Le batching est une technique d'optimisation très intéressante vu qu'elle permet de réduire l'expansion du chiffré (le rapport entre la taille du chiffré et la taille du message en clair), et le nombre d'opérations homomorphes, car plusieurs opérations homomorphes sont faites en parallèle.
Des premiers résultats fixant le nombre de batches (nombre maximum de messages que l'on peut stocker dans un seul chiffré) à 3 montrent que nous obtenons un facteur d'accélération d'environ 2,4 si nous comparons une solution sans batching et une solution avec batching.
Des travaux sont en cours dans cette direction, ils visent à confirmer l'intérêt du batching en donnant des résultats concrets qui évaluent le surcoût du batching et son impact sur les durées des calculs.

IV.4. Conclusion

Dans ce chapitre, nous avons présenté une implantation matérielle d'une deuxième primitive utilisée dans la construction des schémas homomorphes. Cette primitive est l'échantillonnage gaussien discret. Nous avons suivi le même flot de synthèse de haut niveau que celui décrit dans le chapitre III, qui nous offre une flexibilité notamment au niveau de la configuration des paramètres du circuit mais aussi un grand choix au niveau des solutions générées. Des résultats d'implantation sur FPGA permettant de produire des échantillons d'une gaussienne discrète ont été présentés. Dans une deuxième partie, nous avons proposé une accélération de deux primitives du schéma homomorphe FV en se basant sur notre multiplieur polynomial modulaire et notre échantillonneur gaussien discret.

Conclusion générale et perspectives

Considéré comme le Saint Graal de la cryptographie, l'existence d'un schéma de chiffrement permettant la manipulation publique des données chiffrées n'a été prouvée possible qu'en 2009 en utilisant les réseaux Euclidiens. Il s'agit du chiffrement homomorphe.
Nos études ont montré que si cette cryptographie nouvelle a le pouvoir théorique d'offrir une telle propriété, l'efficacité n'est aujourd'hui pas toujours celle attendue : des tailles de clés très grandes, des chiffrés pouvant atteindre des millions de bits par bit de message clair, une complexité de calcul remarquable et des temps de calcul très grands.
Dans un but d'amélioration de l'efficacité, ce manuscrit s'intéresse à l'accélération matérielle des schémas de chiffrement homomorphe.
Nous avons proposé trois contributions à ce sujet : une étude détaillée de l'état de l'art, une implantation flexible et générique des deux primitives les plus coûteuses et fréquentes dans la construction des schémas homomorphes basés sur le problème RLWE et l'évaluation de notre approche sur l'un des cryptosystèmes homomorphes les plus récents et les plus efficaces.

La cryptographie homomorphe étant un domaine scientifique récent et en pleine expansion, son état de l'art est instable. On note de nouvelles constructions en nombre de 3 à 4 en moyenne par an avec des améliorations portant sur des axes différents. Côté pratique, on trouve des implantations logicielles et d'autres matérielles, focalisée chacune sur un aspect particulier. Ainsi, nous avons commencé par proposer une analyse détaillée des travaux existants aussi bien théoriques que pratiques. Nous avons montré que les schémas de la deuxième génération (BGV, FV, …) sont au centre des travaux de la communauté scientifique avec un intérêt pour les schémas de la troisième génération qui ont une meilleure gestion du bruit des chiffrés. Par ailleurs, nous avons testé et reporté les performances de plusieurs implantations logicielles libres pouvant guider le concepteur dans son choix de schémas ou de bibliothèques logicielles en fonction de son application.

Ensuite, j'ai présenté une approche générique et flexible visant à accélérer deux primitives, à savoir la multiplication polynomiale modulaire et l'échantillonnage gaussien discret. Cette approche est basée sur la synthèse de haut niveau. J'ai montré que la synthèse de haut niveau appliquée à la cryptographie homomorphe permet d'obtenir rapidement des circuits compétitifs. Elle peut s'adapter à des contraintes spécifiques telles que la manipulation de grands opérandes, l'évolution rapide de la cryptographie homomorphe (surtout au niveau du paramétrage et dimensionnement) et le compromis latence/ressources fixé par le concepteur. L'approche proposée est également compatible avec des optimisations comme le "batching".

Enfin, nous avons conclu nos travaux par une évaluation de notre approche sur le schéma FV qui semble être l'un des schémas les plus prometteurs à l'heure actuelle. Et nous avons reporté des pistes d'améliorations.

Une première perspective est l'intégration de nos accélérateurs matériels dans un schéma de chiffrement homomorphe complet dans un scénario applicatif donné. Grâce à la flexibilité et à la généricité de notre approche, la configuration des différents paramètres et l'adaptation des accélérateurs aux primitives {génération des clés, chiffrement, déchiffrement, évaluation homomorphe} ne doit pas poser de problème. En revanche, il faut accorder un intérêt particulier à la partie adaptation des primitives si on veut tenir compte du bootstrapping ou du batching par exemple. Les circuits que nous avons proposé peuvent aussi s'adapter à la cryptographie à base de réseaux euclidiens en pleine expansion ces dernières années et base de plusieurs schémas de chiffrement symétrique et de signature numérique, orientés vers les perspectives "post-quantiques".

De plus, une extension de l'application du RNS à un niveau plus haut (d'autres calculs, pas seulement au niveau du produit des coefficients) dans le cadre d'une implantation matérielle pourrait avoir des résultats intéressants. Les travaux de [BEHZ16], qui ont présenté une implantation purement logicielle et optimisée du schéma FV basée sur RNS et ont obtenu une amélioration significative des performances, justifient cette perspective.

En outre, la conception de haut niveau (HLS) appliquée à la cryptographie homomorphe s'avère une bonne alternative aux méthodes de conception classique. Elle s'associe au défi relevé par la dérivation des paramètres concrets définissant des primitives homomorphes efficaces et sûres. Le choix des paramètres est d'ailleurs une question d'une très grande importance qui pour l'instant est traitée au cas par cas, protocole par protocole. Un modèle unifiant les schémas homomorphes permettrait une meilleure exploration de la cryptographie homomorphe.

Enfin, nous terminons ces perspectives par une réflexion qui porte sur le calcul approché ("*approximate computing*") et le chiffrement homomorphe. En effet, le chiffrement homomorphe est basé sur la notion d'erreur ou de bruit qui ne doit pas dépasser un certain seuil pour assurer le déchiffrement. Le calcul approché est un paradigme dans lequel les opérations sont exécutées avec la précision adéquate plutôt que maximale, avec l'objectif d'accélérer le traitement ou de réduire le besoin en énergie. Si le bruit associé aux chiffrés fait partie de ce bruit, nous pourrons faire des calculs approchés à travers la couche de chiffrement sur les données chiffrées. Les résultats actuels dans l'un ou l'autre des domaines montrent que l'on est encore loin de considérer cette alternative dans un scénario applicatif donné. Par exemple, la technique de gestion de bruit et sa remise à un niveau constant doit être étudiée pour assurer que les calculs approchés sont exploitables. Les circuits de calcul approché efficaces existants sont dédiés à des opérandes de petite taille, ce qui n'est pas adéquat avec les schémas homomorphes existants. Je pense toutefois que cette piste est à explorer dans les travaux futurs.

Bibliographie

[AB09] Sanjeev Arora and Boaz Barak. Computational Complexity - A Modern Approach. *Cambridge University Press*, 2009.

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. *In Henri Gilbert, editor, EUROCRYPT 2010, volume 6110 of LNCS, pages 553-572, French Riviera, May 30 - June 3, 2010*. Springer, Berlin, Germany .

[ABD16] Martin Albrecht, Shi Bai, Léo Ducas. A Subfield Lattice Attack on Overstretched NTRU Assumptions - Cryptanalysis of some FHE and Graded Encoding Schemes. *36th Annual International Cryptography Conference, Crypto 2016, Santa Barbara, CA, USA, pages 153-178, August 14-18*, 2016.

[ABG+16a] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, et al.. NFLlib: NTT-based Fast Lattice Library. *RSA Conference Cryptographers' Track , Feb 2016, San Francisco, United States*. 2016.

[ABG+16b] NFLlib library, https://github.com/CryptoExperts/FV-NFLlib

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). *In Proc. of STOC,* pages 99-108, ACM, 1996.

[AMFF+13] C. Aguilar-Melchor, S. Fau, C. Fontaine, G. Gogniat, and R. Sirdey. Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain. *Signal Processing Magazine, IEEE,* 30(2), pages 108–117, 2013.

[APS15] Martin R. Albrecht, Rachel Player and Sam Scott. On the concrete hardness of Learning with Errors, Cryptology ePrint Archive: Report 2015/046, https://eprint.iacr.org/2015/046.

[ASP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. *Cryptology ePrint Archive, Report 2014/094,* 2014. http://eprint.iacr.org/.

[Ben94] Josh Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, 1994.

[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/

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. *Cryptology ePrint Archive, Report 2014/356.*

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS 2012*, pages 309–325, Cambridge, Massachusetts, 2012. ACM.

[Big14] Karim Bigou. Etude théorique et implantation matérielle d'unités de calcul en représentation modulaire des nombres pour la cryptographie sur courbes elliptiques. *Université Rennes 1, 2014. https://tel.archives-ouvertes.fr/tel-01127639.*

[BLLN13] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *Cryptography and Coding*, volume 8308 of *LNCS*, pages 45–64. Springer Berlin Heidelberg, 2013.

[BLN14] Joppe W. Bos, Kristin Lauter, and Michael Naehrig. Private Predictive Analysis on Encrypted Medical Data. *Cryptology ePrint Archive, Report 2014/336,* 2014. http://eprint.iacr.org/

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. On the classical hardness of learning with errors. *In Proc. of STOC*, pages 575-584. ACM, 2013.

[BM04] Jean-Claude Bajard, Jean-michel Muller .Calcul et arithmétique des ordinateurs. Hermès, traité IC2, Lavoisier 2004.

[Bou07] N. Bourbaki, Algèbre : Chapitres 4 à 7, *Masson, 1981 (réimprimé 2007) (1ère éd. (Hermann) 1950-1952*), 422 pages.

[Boy10] Xavier Boyen. Lattice mixing and vanishing trapdoors : A framework for fully secure short signatures and more. *In Phong Q. Nguyen and David Pointcheval, editors, PKC 2010, volume 6056 of LNCS, pages 499 - 517, Paris, France, May 26-28, 2010*. Springer, Berlin, Germany.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer Berlin Heidelberg, 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:109, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology, CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer Berlin Heidelberg, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, pages 1–12, New York, NY, USA, 2014.

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013*. Proceedings, pages 315-335, 2013.

[CDPR16] R. Cramer, L. Ducas, C. Peikert, and O. Regev. Recovering short generators of principal ideals in cyclotomic rings. In *M. Fischlin and J.-S. Coron, editors, Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of Lecture Notes in Computer Science, pages 559–585. Springer, 2016.

[CDS15] Sergiu Carpov, Paul Dubrulle, and Renaud Sirdey. Armadillo : A compilation chain for privacy preserving applications. In *Proceedings of the 3rd International Workshop on Security in Cloud Computing, ASIACCS '15, Singapore, Republic of Singapore,* April 14, 2015, pages 13-19, 2015.

[CG15] Yao Chen and Guang Gong. Integer arithmetic over ciphertext and homomorphic data aggregation. In *Proceedings of 2015 IEEE Conference on Communications and Network Security, pages. 628–632*. IEEE, Piscataway, NJ (2015).

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption : Bootstrapping in less than 0.1 seconds. *IACR Cryptology ePrint Archive, 2016:870, 2016.* http://eprint.iacr.org/2016/870.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *In Advances in Cryptology – EUROCRYPT'97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 103-118, 1997.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *In Henri Gilbert, editor, EUROCRYPT 2010, volume 6110 of LNCS, pages 523-552, French Riviera, May 30 - June 3, 2010*. Springer, Berlin, Germany.

[CLP+17a] Hao Chen, Kim Laine, and Rachel Player. Simple Encrypted Arithmetic Library - SEAL v2.2, techreport, Microsoft 2017.

[CLP+17b] Seal : Simple Encrypted Arithmetic Library, https://sealcrypto.codeplex.com/

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In Hugo Krawczyk, editor, *Public-Key Cryptography – PKC 2014*, volume 8383 of *LNCS*, pages 311–328. Springer Berlin Heidelberg, 2014.

[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, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *LNCS*, pages 487–504. Springer Berlin Heidelberg, 2011.

[CN12] Yuanmi Chen and PhongQ. Nguyen. Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology, EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 502–519. Springer Berlin Heidelberg, 2012.

[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, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 446–464. Springer Berlin Heidelberg, 2012.

[CP16a] Eric Crockett and Chris Peikert. $\Lambda \circ \lambda$: A Functional Library for Lattice Cryptography. https://web.eecs.umich.edu/~cpeikert/pubs/lol.pdf

[CP16b] $\Lambda \circ \lambda$ library, https://github.com/cpeikert/Lol/

[CS15] Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 513-536, 2015.

[CS15a] Anamaria Costache and Nigel P. Smart. Which Ring Based Somewhat Homomorphic Encryption Scheme is Best? *Cryptology ePrint Archive, Report 2015/889,* 2015. http://eprint.iacr.org/

[CT12] Jean-Sébastien Coron and Mehdi Tibouchi. An implementation of the DGHV fully homomorphic scheme, 2012. Available under the GNU General Public License version 2 at https://github.com/coron/fhe.

[DDL+13] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In R. Canetti and J. A. Garay, editors, CRYPTO 2013, Part I, volume 8042 of LNCS, pages 40-56, Santa Barbara, CA, USA, Aug. 18-22, 2013. Springer, Berlin, Germany.

[DG14] Nagarjun C. Dwarakanath and Steven D. Galbraith. Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. *Applicable Algebra in Engineering, Communication and Computing, 25(3): 159-180,* 2014.

[DGL+16] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig and John Wernsing. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. *Microsoft Report*, 2016.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Public Key Cryptography*, volume 1992 of *LNCS*, pages 119–136. Springer Berlin Heidelberg, 2001.

[DM15] Léo Ducas and Daniele Micciancio. FHEW : bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings*, Part I, pages 617-640, 2015.

[DNF+15] Donald D. Chen, Nele Mentens, Frederik Vercauteren, Sujoy Sinha Roy, Ray C. C. Cheung, Derek Pao and Ingrid Verbauwhede. High-Speed Polynomial Multiplication Architecture for Ring-LWE and SHE Cryptosystems. in *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(1): 157-166, Jan. 2015.

[DOS15] Yarkin Doröz, Erdinç Öztürk and Berk Sunar. Accelerating Fully Homomorphic Encryption in Hardware. in *IEEE Transactions on Computers*, vol. 64, no. 6, pages 1509-1521, June 1 2015.

[DOS+15] Yarkin Doröz, Erdinç Öztürk, Erkay Savas and Berk Sunar . Accelerating LTV Based Homomorphic Encryption in Reconfigurable Hardware. In *Güneysu T., Handschuh H. (eds) Cryptographic Hardware and Embedded Systems - CHES 2015. Lecture Notes in Computer Science, vol 9293.* Springer, Berlin, Heidelberg.

[DSES14] Yarkin Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. Toward practical homomorphic evaluation of block ciphers using Prince. *2014. WAHC'14 - 2nd Workshop on Applied Homomorphic Cryptography and Encrypted Computing*.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *In GeorgeRobert Blakley and David Chaum, editors, Advances in Cryptology,* volume 196 of LNCS, pages 10–18. Springer Berlin Heidelberg, 1985.

[FG07] Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP J. Inf. Secur.*, 2007:15:1–15:15, January 2007.

[FSF+13] Simon Fau, Renaud Sirdey, Caroline Fontaine, Carlos Aguilar Melchor, and Guy Gogniat. Towards practical program execution over fully homomorphic encryption schemes. In Eighth *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2013, Compiegne, France, October 28-30, 2013*, pages 284-290, 2013.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive, Report 2012/144*, 2012. http://eprint.iacr.org/.

[Gal02] Steven D. Galbraith. Elliptic curve paillier schemes. *Journal of Cryptology*, 15(2):129–138, 2002.

[Gen09a] Craig Gentry. A fully homomorphic encryption scheme. *PhD thesis, Stanford University, 2009*. crypto.stanford.edu/craig.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. *In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169-178, 2009.

[Gen14] Craig Gentry. Computing on the edge of chaos : Structure and randomness in encrypted computation. *Cryptology ePrint Archive, Report 2014/610, 2014*. http://eprint.iacr.org/2014/610.

[GH11] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology - EUROCRYPT 2011, 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 129-148, 2011.

[GH11a] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 107–109, Washington, DC, USA, 2011. IEEE Computer Society.

[GH11b] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology*, EUROCRYPT'11, pages 129–148, Berlin, Heidelberg, 2011. Springer-Verlag.

[GHR+14] C. Gentry, S. Halevi, M. Raykova, and D. Wichs. Garbled ram revisited, part I. *Cryptology ePrint Archive*, Report 2014/082, 2014.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 850-867, 2012.

[GHS15] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit (Updated Implementation). *Cryptology ePrint Archive, Report 2012/099*, 2012. http://eprint.iacr.org/

[GLN12] Thore Graepel, Kristin Lauter, Michael Naehrig. ML Confidential: Machine Learning on Encrypted Data. *In: Kwon T., Lee MK., Kwon D. (eds) Information Security and Cryptology – ICISC 2012*. Lecture Notes in Computer Science, vol 7839. Springer, Berlin, Heidelberg.

[GM84] Shai Goldwasser and Silvio Micali .Probabilistic encryption. *J. Comput. Syst. Sci.,28(2)* :270-299, 1984.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. *In Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC '08)*. ACM, New York, NY, USA, 197-206.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *LNCS*, pages 75–92. Springer Berlin Heidelberg, 2013.

[Gue14] Antoine Guellier. Can Homomorphic Cryptography ensure Privacy?. *[Research Report] RR-8568, IRISA, Supélec Rennes, équipe Cidre, Array*. 2014, pp.109.

[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.

[HG01] Nick Howgrave-Graham. Approximate Integer Common Divisors. In *Silverman J.H. (eds) Cryptography and Lattices*, Lecture Notes in Computer Science, vol. 2146, 2001, Springer, Berlin, Heidelberg.

[HKR+16] James Howe, Ayesha Khalid, Ciara Rafferty, Francesco Regazzoni, Maire O'Neill. On Practical Discrete Gaussian Samplers For Lattice-Based Cryptography. *IEEE Transactions on Computers*. 2016.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Joe Buhler, editor, ANTS, volume 1423 of Lecture Notes in Computer Science, pages 267–288*. Springer, 1998.

[HS14] Shai Halevi and Victor Shoup. Algorithms in helib. In *Advances in Cryptology-CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 554-571, 2014.

[HS15] Shai Halevi and Victor Shoup. Bootstrapping for helib. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 641-670, 2015.

[JCM+15] Cedric Jayet-Griffon, M.-A. Cornelie, P. Maistri, Ph. Elbaz-Vincent and R. Leveugle. Polynomial Multipliers for Fully Homomorphic Encryption on FPGA. In *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Mexico City, 2015, pages 1-6.

[KL15] Miran Kim and Kristin Lauter. Private Genome Analysis through Homomorphic Encryption. *Cryptology ePrint Archive, Report 2015/965, 2015*.

[KY76] D. E. Knuth and A. C. Yao. The complexity of non uniform random number generation. In J.F. Traub (Ed) Algorithms and complexity: new directions and recent results, pages 357– 428, 1976.

[Lep14] Tancrède Lepoint. A proof-of-concept implementation of the homomorphic evaluation of SIMON using FV and YASHE leveled homomorphic cryptosystems, 2014. https://github.com/tlepoint/homomorphic-simon.

[LLN14] Kristin E. Lauter, Adriana López-Alt and Michael Naehrig. Private Computation on Encrypted Genomic Data. In *Aranha D., Menezes A. (eds) Progress in Cryptology LATINCRYPT 2014. LATINCRYPT 2014. Lecture Notes in Computer Science, vol 8895*, pages 3-27. Springer, Cham.

[LLS14] Fabien Laguillaumie, Adeline Langlois, Damien Stehlé. Chiffrement avancé à partir du problème Learning With Errors. *Sylvain PEYRONNET. Informatique Mathématique : une photographie en 2014, Presses universitaires de Perpignan*, 2014, 9782354122287.

[LLS+16] Yehuda Lindell and Nigel P. Smart and Eduardo Soria-Vazquez. More Efficient Constant-Round Multi-Party Computation from BMR and SHE. *Cryptology ePrint Archive, Report 2016/156, 2016*. http://eprint.iacr.org/

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In *Progress in Cryptology - AFRICACRYPT 2014,7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, pages 318-335, 2014.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Kiayias A. (eds) Topics in Cryptology – CT-RSA 2011. CT-RSA 2011. Lecture Notes in Computer Science, vol 6558*, pages 319–339, Springer, Berlin, Heidelberg.

[LP13] Tancrède Lepoint and Pascal Paillier. On the minimal number of bootstrappings in homomorphic circuits. In *Financial Cryptography and Data Security - FC 2013 Workshops, USEC and WAHC 2013, Okinawa, Japan, April 1, 2013, Revised Selected Papers*, pages 189-200, 2013.

[LP16] Kim Laine and Rachel Player. Simple encrypted arithmetic library - seal (v2.0). *Technical report, Microsoft Research*, September 2016.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013.

[LTV12] Adriana Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-y multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC-2012, New York, NY, USA, May 19 - 22, 2012*, pages 1219-1234, 2012.

[MBF16] Vincent Migliore, Guillaume Bonnoron and Caroline Fontaine. Determination and exploration of practical parameters for the latest Somewhat Homomorphic Encryption (SHE) Schemes. Working paper, 2016.

[MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.* 37, 1 (April 2007), 267-302.

[MRL+15] Vincent Migliore, Maria Mendez Real, Vianney Lapotre, Arnaud Tisserand, Caroline Fontaine and Guy Gogniat. Exploration of polynomial multiplication algorithms for homomorphic encryption schemes. In *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Mexico City, 2015, pages 1-6.

[MRL+16a] V. Migliore, M. M. Real, V. Lapotre, A. Tisserand, C. Fontaine and G. Gogniat. Fast polynomial arithmetic for Somewhat Homomorphic Encryption operations in hardware with Karatsuba algorithm. *2016 International Conference on Field-Programmable Technology (FPT)*, Xi'an, 2016, pages. 209-212.

[MRL+16b] V. Migliore; M. Mendez Real, V. Lapotre, A. Tisserand, C. Fontaine, G. Gogniat. Hardware/Software co-Design of an Accelerator for FV Homomorphic Encryption Scheme using Karatsuba Algorithm. In *IEEE Transactions on Computers, vol.PP, no.99*, pp.1-1.

[MRS88] S. Micali, C. Rackoff, and R. Sloan. The Notion of Security for Probabilistic Cryptosystems. *SIAM Journal on Computing, 17(2):412–426*, April 1988. Special issue on cryptography.

[NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Christian Cachin and Thomas Ristenpart, editors, CCSW,* pages 113–124. ACM, 2011.

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM conference on Computer and communications security*, CCS '98, pages 59–66, New York, NY, USA, 1998.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *LNCS*, pages 308–318. Springer Berlin Heidelberg, 1998.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. *In Proceedings of the 17th international conference on Theory and application of cryptographic techniques,* EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.

[PBS11a] Henning Perl, Michael Brenner, and Matthew Smith. Poster : an implementation of the fully homomorphic smart-vercauteren crypto-system. In *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21*, 2011, pages 837-840, 2011.

[PBS11b] hcrypt-project : libScarab library , https://github.com/hcrypt-project/libScarab.

[PNP+15] Thomas Pöppelmann, Michael Naehrig, Andrew Putnam and Adrián Macías. Accelerating Homomorphic Evaluation on Reconfigurable Hardware. In *Güneysu T., Handschuh H. (eds) Cryptographic Hardware and Embedded Systems -- CHES 2015. Lecture Notes in Computer Science, vol 9293.* Springer, Berlin, Heidelberg.

[PV15] Marie Paindavoine and Bastien Vialla. Minimizing the number of bootstrappings in fully homomorphic encryption. In *Selected Areas in Cryptography - SAC 2015, 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pages 25-43, 2015.

[RAD78] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, *Academia Press*, pages 169-179, 1978.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *In Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing,* STOC '05, pages 84–93, New York, NY, USA, 2005.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. J. ACM, 56(6), 2009.

[RJV+15] Sujoy Sinha Roy, Kimmo Järvinen, Frederik Vercauteren, Vassil S. Dimitrov and Ingrid Verbauwhede. Modular Hardware Architecture for Somewhat Homomorphic Function Evaluation. In *Güneysu T., Handschuh H. (eds) Cryptographic Hardware and Embedded Systems -- CHES 2015. CHES 2015. Lecture Notes in Computer Science, vol 9293.* Springer, Berlin, Heidelberg.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In *Yuval Ishai, editor, Theory of Cryptography*, volume 6597 of LNCS, pages 219–234. Springer Berlin Heidelberg, 2011.

[S+17] W. A. Stein et al. Sage Mathematics Software (Version 8.0). The Sage Development Team, 2017. http://www.sagemath.org.

[Sav97] John E. Savage, Models of Computation: Exploring the Power of Computing, *Addison-Wesley Longman Publishing Co.,* Inc., Boston, MA, 1997.

[Saa15] M.-J. O. Saarinen. Gaussian sampling precision and information leakage in lattice cryptography. Cryptology ePrint Archive, Report 2015/953, 2015.

[SCL+17] Gizem S. Cetin, Hao Chen, Kim Laine, Kristin E. Lauter, Peter Rindal and Yuhou Xia. Private Queries on Encrypted Genomic Data. *Cryptology ePrint Archive, Report 2017/207,* 2017. http://eprint.iacr.org/

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 377–394. Springer Berlin Heidelberg, 2010.

[SS11] Damien Stehlé, Ron Steinfeld. Making NTRU as Secure as Worst-Case Problems over Ideal Lattices. In *Paterson K.G. (eds) Advances in Cryptology – EUROCRYPT 2011. Lecture Notes in Computer Science, vol 6632*, pages 27-47. Springer, Berlin, Heidelberg.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Proceedings of the 13th international conference on Practice and Theory in Public Key Cryptography*, PKC'10, pages 420–443, Paris, France, 2010.

[TJW+16] Haixu Tang, Xiaoqian Jiang, Xiaofeng Wang,Shuang Wang, Heidi Sofia,Dov Fox, Kristin Lauter and al.. Protecting genomic data analytics in the cloud: state of the art and opportunities. *BMC Med Genomics. 2016, pages 9- 63.* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5062944/

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology, EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer Berlin Heidelberg, Gilbert, Henri edition, 2010.

[Wen15] Lu Wenjie. Ring-learning with errors and HElib. *SlideShare, 2015*, https://www.slideshare.net/ssuser4c5f79/h-elib.

[WYL+12] Wei Wang, Yin Hu, Lianmu Chen, Xinming Huang and Berk Sunar Accelerating fully homomorphic encryption using GPU. In *IEEE Conference on High Performance Extreme Computing, HPEC 2012, Waltham, MA, USA, September 10-12, 2012*, pages 1–5. IEEE (2012).

[WX13] Wei Wang and Xinming Huang. FPGA implementation of a large-number multiplier for fully homomorphic encryption. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013), Beijing, China, May 19-23, 2013*. Pages 2589–2592. IEEE (2013).

[WXN+14] Wei Wang, Xinming Huang, Niall Emmart and Charles Weems. VLSI design of a large-number multiplier for fully homomorphic encryption. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pages 1879-1887, Sept. 2014.

[XCW+17] Chen Xu, Jingwei Chen, Wenyuan Wu and Yong Feng. Homomorphically Encrypted Arithmetic Operations over the Integer Ring. *Cryptology ePrint Archive, Report 2017/387,* 2014. http://eprint.iacr.org/

Publications de l'auteure

Conférences internationales:

A. Mkhinini, P. Maistri, R. Leveugle and R. Tourki, HLS design of a hardware accelerator for Homomorphic Encryption, IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS'17), pp. 178-183, Dresden, Allemagne, 19 au 21 avril 2017.

Mkhinini A., Maistri P., Leveugle R., Tourki R., Machhout M., "A flexible RNS-based large polynomial multiplier for Fully Homomorphic Encryption", 11th IEEE International Design & Test Symposium (IDT'16), pp. 131-136, Hammamet, TUNISIA, 18 au 20 décembre 2016.

Conférences nationales :

Mkhinini A., Maistri P., Leveugle R., "Chiffrement homomorphique : entre développements théoriques et implantations pratiques", 8ème Colloque du GdR SoC-SiP, Paris, FRANCE, 11 au 13 juin 2014.

Annexe A. Exemples de paramètres pratiques pour des schémas homomorphes basés sur RLWE

Dans cette annexe, je présente un ensemble de paramètres pratiques pour les schémas homomorphes les plus efficaces. Ces paramètres sont extraits des travaux les plus récents [APS15, CS15, MBF16 et BF16] qui étudient le problème de la dérivation des paramètres pratiques des schémas homomorphes basés sur le problème RLWE répondant au compromis efficacité-sécurité-fonctionnalité. Des tests sur des implantations logicielles existantes ont été effectués pour valider certaines configurations.
Je résume dans la suite des listes de combinaisons possibles de n, q et L en jouant à chaque fois sur un ou plusieurs paramètres. Les tableaux présentés pourront guider le concepteur dans la définition de son cahier de charge en se basant sur les contraintes de son application.

Tableau A.1. Exemples de paramètres pratiques pour le schéma homomorphe FV permettant d'atteindre un niveau de sécurité de 80 bits

L	$\omega = 27\ bits$		$\omega = 32\ bits$		$\omega = 64\ bits$	
	n	$log_2 q$	n	$log_2 q$	n	$log_2 q$
1	1040	49	1021	54	1721	87
5	3538	154	3167	157	3884	191
6	4179	182	3664	187	5120	221
7	4894	210	4334	215	5782	250
8	5579	239	5002	244	6491	279
9	6250	268	5677	273	7183	308
10	6957	298	6082	298	6800	333
15	10520	449	9138	448	9872	482
20	14156	606	12246	602	12931	636

Tableau A.2. Exemples de paramètres pratiques pour le schéma homomorphe FV permettant d'atteindre un niveau de sécurité de 128 bits

L	$\omega = 27\ bits$		$\omega = 32\ bits$		$\omega = 128\ bits$	
	n	$log_2 q$	n	$log_2 q$	n	$log_2 q$
1	1660	50	1935	54	2184	87
5	5754	160	5881	161	6190	198
6	6848	190	7044	195	8263	229
7	7976	220	8109	225	9491	259
8	9071	250	9247	255	10569	289
9	10213	280	10450	286	11661	320
10	11309	311	11675	317	12844	351
15	17259	470	12311	460	13225	494
20	23189	633	16572	619	17477	653

Tableau A.3. Exemples de paramètres pratiques pour le schéma homomorphe Yashe permettant d'atteindre un niveau de sécurité de 80 bits pour un $\boldsymbol{\omega = 32\ bits}$

L	n	$log_2 q$
1	1641	64
3	3329	128
5	5081	192
7	6707	256
15	16463	512
31	26974	1024

Tableau A.4. Exemples de paramètres pratiques pour le schéma homomorphe BGV permettant d'atteindre un niveau de sécurité de 80 bits

L	n	$log_2 q$
2	793	14
5	10213	271
10	25486	274
20	51003	275
30	101853	278

TITRE : Implantation matérielle de chiffrements homomorphiques

RESUME Une des avancées les plus notables de ces dernières années en cryptographie est sans contredit l'introduction du premier schéma de chiffrement complètement homomorphe par Craig Gentry. Ce type de système permet de réaliser des calculs arbitraires sur des données chiffrées, sans les déchiffrer. Cette particularité permet de répondre aux exigences de sécurité et de protection des données, par exemple dans le cadre en plein développement de l'informatique en nuage et de l'internet des objets. Les algorithmes mis en œuvre sont actuellement très coûteux en temps de calcul, et généralement implantés sous forme logicielle. Les travaux de cette thèse portent sur l'accélération matérielle de schémas de chiffrement homomorphes. Une étude des primitives utilisées par ces schémas et la possibilité de leur implantation matérielle est présentée. Ensuite, une nouvelle approche permettant l'implantation des deux fonctions les plus coûteuses est proposée. Notre approche exploite les capacités offertes par la synthèse de haut niveau. Elle a la particularité d'être très flexible et générique et permet de traiter des opérandes de tailles arbitraires très grandes. Cette particularité lui permet de viser un large domaine d'applications et lui autorise d'appliquer des optimisations telles que le batching. Les performances de notre architecture de type co-conception ont été évaluées sur l'un des cryptosystèmes homomorphes les plus récents et les plus efficaces. Notre approche peut être adaptée aux autres schémas homomorphes ou plus généralement dans le cadre de la cryptographie à base de réseaux.

Mots clés : Systèmes intégrés numériques, systèmes sécurisés, chiffrement homomorphique, synthèse de haut niveau.

TITLE: Hardware implementation of homomorphic encryption schemes

ABSTRACT One of the most significant advances in cryptography in recent years is certainly the introduction of the first fully homomorphic encryption scheme by Craig Gentry. This type of cryptosystem allows performing arbitrarily complex computations on encrypted data, without decrypting it. This particularity allows meeting the requirements of security and data protection, for example in the context of the rapid development of cloud computing and the internet of things. The algorithms implemented are currently very time-consuming, and most of them are implemented in software. This thesis deals with the hardware acceleration of homomorphic encryption schemes. A study of the primitives used by these schemes and the possibility of their hardware implementation is presented. Then, a new approach allowing the implementation of the two most expensive functions is proposed. Our approach exploits the high-level synthesis. It has the particularity of being very flexible and generic and makes possible to process operands of arbitrary large sizes. This feature allows it to target a wide range of applications and to apply optimizations such as batching. The performance of our co-design was evaluated on one of the most recent and efficient homomorphic cryptosystems. It can be adapted to other homomorphic schemes or, more generally, in the context of lattice-based cryptography.

Keywords: Digital integrated systems, secured systems, homomorphic encryption, high level synthesis.

INTITULE ET ADRESSE DU LABORATOIRE

Laboratoire TIMA, 46 avenue Félix Viallet, 38031 Grenoble Cedex, France.