Méthodologie de conception et d'utilisation d'algorithme d'intelligence artificielle à partir de données à caractère sensible : étude et application à la reconnaissance acoustique sous-marine

Tom Laborde

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Mathématiques*

Par
Tom LABORDE

Méthodologie de conception de modèles d'apprentissage automatique sur données à caractère confidentiel

Étude et application à un cas d'usage industriel

Thèse présentée et soutenue à l'Université de Rennes 1, le 13 décembre 2023
Unité de recherche : IRMAR - Institut de recherche mathématique de Rennes

Rapporteurs avant soutenance :

Srecko BRLEK	Professeur des universités - Université du Québec
Patrick PEREZ	Directeur scientifique - Valeo

Composition du Jury :

Président :	Sylvain DUQUESNE	Professeur des universités	Université de Rennes 1
Examinateurs :	Fanny DELEBECQUE	Maître de conférence	Université de Toulouse
	Mohammed LEMOU	Directeur de recherche	CNRS en disponibilité chez Ravel Technologies
	Marie-Hélène VIGNAL	Maître de conférence	Université de Toulouse
Dir. de thèse :	Florian MÉHATS	Professeur des universités	Université de Rennes 1 en disponibilité chez Ravel Technologies
Co-dir. de thèse :	Alexandre GENSSE	Ingénieur en data-science	Naval Group

Acknowledgement

Je souhaite exprimer ma gratitude envers les membres de l'équipe MINGUS : Philippe CHARTIER, Florian MEHATS, Mohammed LEMOU. Travailler avec eux a été pour moi un privilège et je garderai toujours en mémoire les conseils qu'ils m'ont donnés, ainsi que ce que j'ai pu observer et apprendre à partir de leurs méthodes de travail. Je retiendrai notamment la rigueur, l'efficacité et l'humilité qui siéent à ces chercheurs de très haut niveau. Je tiens aussi à les remercier pour leur simplicité et leur accessibilité, au-delà de leurs compétences scientifiques, j'ai pu rencontrer des personnes très sympathiques et bienveillantes. En particulier, je remercierai mon directeur de thèse Florian, pour la patience et la pédagogie dont il a su faire preuve envers moi, Mohammed pour son aide technique précieuse et Philippe pour ses avis éclairés.

Je remercie aussi mes rapporteurs Srecko BRLEK et Patrick PEREZ qui ont pris la peine de lire en détail mon manuscrit et ont accepté cette soutenance ainsi que Fanny DELEBECQUE, Sylvain DUQUESNE et Marie-Hélène VIGNAL. Merci pour votre temps et vos commentaires constructifs qui m'ont permis de faire avancer mon travail.

Je souhaite également remercier l'ensemble de l'équipe CEMIS avec qui j'ai eu le plaisir d'évoluer au quotidien ces trois dernières années. Je ne pourrais pas citer de manière exhaustive l'ensemble des personnes qui m'ont apporté leur soutien, en effet, elles sont très nombreuses. Chacun se reconnaitra, bien entendu.

Je tiens également à remercier ma famille et mes deux meilleurs amis Ousmane et Pierre qui m'ont toujours soutenu lors de chaque étape de mon parcours. Ces personnes me connaissent et savent à quel point elles comptent pour moi. Je tiens cependant à les remercier pour leur indulgence et leur compréhension quant à mon indisponibilité lors des périodes de rush.

Pour conclure, il m'est impossible de ne pas remercier Alexandre GENSSE, mon encadrant industriel. Alexandre a toujours su être là pour moi, n'hésitant pas à prendre sur son temps personnel pour que nous atteignions nos objectifs. Je ne compte plus le nombre de discussions et de débats scientifiques que nous avons pu avoir. Son investissement dans cette thèse va bien au-delà de l'investissement standard attendu pour un encadrant industriel. Sa pertinence technique et son professionnalisme ont été d'une grande aide ces

trois années durant. Une thèse est une aventure scientifique mais aussi humaine et je suis heureux d'avoir pu la vivre.

SOMMAIRE

Liste des abréviations

ACP : Analyse en Composantes Principales
ABE : Attribute-Based Encryption
AD-SIM : Adaptive SIM Secure
AES : Advanced Encryption Standard
ASM : Acoustique Sous-Marine
CC : Cloud Computing
CCPA : California Consumer Privacy Act
CNN : Convolutional Neural Network
DCR : Decisional Composite Residuosity
DDH : Hypothèse décisionnelle de Diffie-Hellman
DFT : Discrete Fourier Transform
ES : Émission Sonar
FE : Functional Encryption
GDPR : General Data Protection Regulation
GGM : Generic Group Model
GMP : GNU Multiple Precision Arithmetic Library
GPU : Graphics Processing Unit
HE : Homomorphic Encryption
HIPPA : Health Insurance Portability and Accountability Act
IA : Intelligence Artificielle
IBE : Identity-Based Encryption
IND : Indistinguable secure
IPE : Inner Product Encryption
IPFE : Inner Product Functional Encryption
LDA : Linear Discriminant Analysis
LWE : Learning With Errors
ML : Machine Learning
MNIST : Modified ou Mixed National Institute of Standards and Technology
MPC : secure Multi-Party Computation

NIST : Institut National de Standard et Technologie
PE : Predicate Encryption
PPML : Privacy-Preserving Machine Learning
PPT : Probabilistic Polynomial-Time
RLWE : Ring Learning With Errors
RN : Réseau de Neurones
SIM : Subscriber Identity Module
SIS : Short Integer Solution
SVM : Support Vector Machine
SVP : Shortest Vector Problem
TF : Transformée de Fourier

Chapitre 1

Introduction

L'utilisation du Machine Learning (ML) est devenue courante dans de nombreux domaines d'application. En général, un modèle de ML performant repose sur un grand volume de données d'apprentissage et sur de puissantes capacités de calcul informatique. Cependant, cette exigence pousse de nombreux utilisateurs à externaliser la conception et l'entraînement des modèles à des entreprises spécialisées, que ce soit de manière directe ou via internet (Cloud Computing). Cette pratique soulève des interrogations en matière de protection de la vie privée car elle expose les données confidentielles à des risques de fuite. De plus, l'évolution des réglementations restreignant l'accès et l'utilisation de données confidentielles complique l'exploitation optimale de la puissance du Machine Learning. Enfin, un modèle de ML entraîné peut également être vulnérable aux attaques adverses comme les attaques par inférence, par attributs ou encore les attaques par rétro-ingénierie. C'est pourquoi des solutions bien conçues de ML préservant la vie privée (Privacy Preserved Machine Learning (PPML)) sont nécessaires pour de nombreuses applications émergentes. De plus en plus d'efforts de recherche significatifs de la part des universités et de l'industrie peuvent être observés dans le domaine du PPML qui visent à intégrer des techniques de préservation de la confidentialité dans le pipeline ML ou dans des algorithmes spécifiques. Parmi les différentes approches existantes pour garantir la confidentialité des données, certaines reposent sur des techniques traditionnelles comme l'anonymisation ou le filtrage, tandis que d'autres plus récentes utilisent des crypto systèmes comme le chiffrement homomorphe (Homomorphic Encryption (HE)), le chiffrement fonctionnel (Functional Encryption (FE)) ou le calcul multipartite sécurisé (secure multi-party computation (MPC)).

Dans cette thèse, nous proposons une méthode d'application du chiffrement fonctionnel à des données MNIST ainsi qu'à des données acoustiques sous-marines représentatives de cas d'usages industriels réels.

1.1 Contexte historique

La cryptographie est une discipline qui remonte à l'Antiquité et qui a été développée pour permettre aux agents de communiquer sans être compris par l'ennemi pendant les périodes de conflit. Les premières méthodes de chiffrement étaient basées sur un système de codage et de décodage convenu entre les deux parties avant leur séparation, leur permettant de chiffrer et de déchiffrer les messages qu'ils s'envoyaient par messager une fois séparés. Le chiffrement César est un exemple célèbre de méthode de chiffrement simple utilisée par Jules César pour communiquer avec ses généraux pendant les guerres. Cette méthode consiste à décaler chaque lettre d'un certain nombre de positions dans l'alphabet. Par exemple, avec un décalage de 3, la lettre A devient D, la lettre B devient E, et ainsi de suite. Avec l'avancée des technologies, notamment les moyens de communication comme internet, les besoins en cryptographie ont évolué. En effet, le nombre de communications et d'interlocuteurs a explosé, entraînant une augmentation des besoins en protection, authentification et anonymisation. Les interlocuteurs ne sont plus toujours en mesure de se rencontrer pour convenir d'une clé secrète à l'avance. En effet, deux agents veulent pouvoir échanger une clé secrète en ligne tout en étant sûrs qu'aucun pirate informatique (hacker en anglais) n'interfère ou ne les dupe. Pour répondre à ce besoin, les signatures digitales [102] et le chiffrement à clé publique [36] ont été inventés. Ce dernier est un système permettant à un agent de confiance de générer une clé publique qui autorise tous les utilisateurs à chiffrer des messages. Seul le générateur de la clé publique est alors en mesure de déchiffrer le contenu. L'avènement d'internet a de plus engendré une quantité et une variété de bases de données inédites dans l'histoire. Cette évolution, combinée à l'augmentation des capacités de calcul des ordinateurs a permis le développement de l'intelligence artificielle (IA). En particulier, le ML, un type de modèle basé sur l'apprentissage automatique, et notamment une de ses branche phare, l'apprentissage profond (Deep Learning), ont fait leurs preuves dans de nombreux domaines où ils ont surpassé les approches traditionnelles comme la vision par ordinateur, le traitement du langage naturel, la reconnaissance vocale ou audio [74, 59, 99]. Cependant, ils nécessitent de grandes quantités de données et des capacités de calcul élevées, reposant sur les processeurs graphiques GPU chargés des calculs en parallèle, pour être performants. Le marché s'est adapté à ces nouveaux besoins et de nouvelles infrastructures spécialisées pour le ML ont vu le jour, les principaux acteurs étant Amazon, Microsoft, Google, et IBM. En particulier, ces entreprises mettent à disposition des plateformes permettant à l'utilisateur

de louer leurs capacités de stockage et de calcul (Cloud Computing). Toutefois, cette externalisation des calculs soulève des problèmes de confidentialité des données, notamment dans les secteurs tels que la médecine, la finance ou encore la défense. Des organismes comme la Health Insurance Portability and Accountability Act (HIPPA), la European General Data Protection Regulation (GDPR), la Cybersecurity Law of China, le New York SHIELD Act et le California Consumer Privacy Act (CCPA) ont été mises en place pour limiter l'exploitation des données privées. Cependant, il est difficile de garantir que la confidentialité est entièrement respectée, surtout pour les données très sensibles. Les utilisateurs ne peuvent pas se contenter de faire confiance sans garantie, car il y a eu des cas de violation de données [103, 117]. De plus, il est possible d'effectuer des attaques par inférence sur les modèles de ML, une des plus connues étant la "membership attack" [105, 20, 66, 77, 89] qui consiste à inférer sur les données d'apprentissage. Pour répondre à ces besoins, se développent de nouvelles solutions ayant pour objectif de pouvoir faire du ML tout en préservant la confidentialité (PPML - Privacy-Preserved Machine Learning en anglais). Ce domaine propose des approches traditionnelles basées sur l'anonymisation comme le k-anonyme [108], l-diversité [82] et le t-proximité [78] qui suppriment les informations identifiantes ou quasi-identifiantes des données avant leur exploitation pour l'entraînement. D'autres approches basées sur la ϵ-confidentialité différentielle consistent à ajouter du bruit dans les données afin qu'on ne puisse tirer aucune déduction statistique sur les informations privées [39, 41, 40]. Cependant, les approches les plus récentes, basées sur la cryptographie, ont pour avantages de garantir une meilleure sécurité contre les attaques par inférence ou désanonymisation [122, 98, 105, 97], en particulier contre un adversaire ayant une connaissance a priori sur les données à protéger. Ces méthodes utilisent des crypto-systèmes pour protéger les données d'apprentissage ou les modèles partagés [113, 123, 124, 46, 58, 31, 51, 30, 88]. Parmi ces approches cryptographiques, on distingue différentes familles : d'une part les approches basées sur des chiffrements symétriques, comme la norme de chiffrement avancé (AES) combinée à des circuits brouillés basés sur des protocoles de calcul multipartite sécurisé (MPC) [104, 101], et d'autrepart les nouveaux schémas émergents reposant sur le chiffrement homomorphe [48, 114, 25, 83, 4] ou sur le chiffrement fonctionnel [24, 52, 3, 16, 14, 1, 2]. Ces approches impliquent des calculs sur données chiffrées où seulement une partie possède la clé permettant de déchiffrer et d'accéder aux résultats. Elles ont pour avantage de nécessiter uniquement le transfert de données chiffrées, bien plus simple à mettre en place que les transferts de circuits brouillés des protocoles MPC qui sont beaucoup plus lourds et volumineux. De

nombreuses approches de PPML basées sur ces deux types de schéma de chiffrement ont vu le jour ces dernières années [113, 123, 124, 46, 58, 31, 51, 30, 88, 57, 90, 32, 80]. Dans cette thèse, nous étudierons les approches par FE, ce type de schéma pouvant être vu comme une variante d'un schéma de chiffrement à clé publique permettant de déchiffrer f(x) à la place de x sans donner d'autres informations sur x que f(x). Nous détaillerons cette définition dans la suite du manuscrit.

1.2 Histoire du chiffrement fonctionnel

Le chiffrement fonctionnel (FE) est une méthode de chiffrement issue du chiffrement basé sur l'identité (Identity-Based Encryption (IBE)) [23, 22, 120, 5, 6], une technique de chiffrement à clé publique où chaque utilisateur possède sa propre clé secrète. Dans l'IBE, tous les utilisateurs chiffrent leurs données avec la même clé, mais chaque utilisateur peut uniquement déchiffrer ses propres données avec sa clé secrète, ce qui évite les risques de collusion si plusieurs utilisateurs collaborent. Cette approche évolue ensuite en chiffrement basé sur l'attribut (Attribute-based encryption (ABE)) [55, 93, 56, 76, 119, 53, 26], qui fonctionne selon le même principe de chiffrement à clé publique, mais permet aux utilisateurs de déchiffrer les données en fonction de leurs attributs. Ainsi, les agents de la même catégorie sont en mesure d'accéder aux mêmes informations. Vient ensuite le chiffrement prédicatif (predicate encryption (PE)) [67, 68, 91, 47, 54], qui permet à un utilisateur de chiffrer des données sous une condition prédéfinie, de sorte que seuls les utilisateurs qui satisfont cette condition peuvent déchiffrer les données. Ces approches présentent cependant un problème : elles sont en effet "tout ou rien" (all or nothing), ce qui signifie que l'utilisateur ne peut, soit pas accéder au message souhaité, soit pas le déchiffrer entièrement. C'est dans ce contexte que le concept de chiffrement fonctionnel a été introduit en 2010 par O'Neill [98] et en 2011 par Boneh, Sahai et Waters [32]. Cette nouvelle perspective élargie des systèmes de chiffrement inclut tous les systèmes précédemment mentionnés. Le FE est un type de schéma de chiffrement permettant de déchiffrer un calcul sélectif $f(x)$ du message x chiffré plutôt que le message x lui-même. Chaque clé permet de réaliser un calcul différent du message sans avoir accès à ce dernier. Un chiffrement fonctionnel permettant de calculer n'importe quelle fonction de message souhaitée serait la solution idéale pour répondre aux besoins du ML et du CC. Certaines constructions de ce type ont été proposées, mais elles reposent sur l'existence de primitives très complexes comme l'obfuscation indistinguable [44] ou sur les applications multilinéaires [45]. Ces construc-

tions sont extrêmement inefficaces, au point que même une preuve de concept semble infaisable pour le moment. De plus, toutes les constructions d'applications multilinéaires connues [43, 33, 49], ainsi que certaines constructions reposant sur l'obscurcissement de l'indiscernabilité, ont récemment été cassées.

En 2015, Abdalla, Bourse, De Caro et Pointcheval [1] présentent le premier schéma de FE efficace permettant de calculer une fonction, en l'occurrence un produit scalaire choisi, du message crypté. De manière générale, un schéma de chiffrement du produit scalaire (Inner Product Encryption IPE) comprend plusieurs éléments. Tout d'abord, il contient un algorithme de chiffrement à clé publique qui permet de chiffrer un vecteur message x pour obtenir son chiffré c_x. Ensuite, il comprend un algorithme de génération de clé qui permet de créer une clé fonctionnelle Sk_y à partir d'une clé secrète et d'un vecteur fonction y. Enfin, il inclut un algorithme de déchiffrement qui utilise la clé fonctionnelle Sk_y et le chiffré c_x pour obtenir le produit scalaire $x \cdot y$. A première vue, la fonctionnalité peut sembler limitée, mais en réalité elle trouve déjà de multiples cas d'usages pratiques comme le calcul de la distance de Hamming, de la distance entre deux vecteurs et donc entre deux mots, de statistiques simples comme les moyennes pondérées et même de certains modèles de ML linéaire comme les machines à vecteurs de support (Support Vector Machine SVM) ou les réseaux de neurones à fonction d'activation linéaire. Au niveau de la sécurité, les constructions IPE [1] sont basées sur des hypothèses standards comme l'apprentissage avec erreur (Learning With Error LWE) et l'hypothèse décisionnelle de Diffie-Hellman (Decisional Diffie-Hellman DDH). Elles sont prouvées sécures contre les attaques sélectives, c'est-à-dire contre des adversaires qui doivent choisir leurs inputs (messages m_i d'entrée qui sont ensuite chiffrés) avant le "security game" et ne sont pas en mesure de s'adapter aux précédents tests. Autrement dit, des adversaires n'ayant le droit qu'à une attaque naïve pour tester le schéma. Agrawal and al [18] propose des constructions basées sur les mêmes hypothèses mais résistant cette fois-ci aux attaques adaptatives : les inputs (messages m_i) peuvent être adaptés en fonction des chiffrés obtenus lors des précédentes itérations lors de la phase de challenge. Un peu plus tard, Agrawal a prouvé que les schémas proposés dans [18] sont également "adaptive SIM secure"(AD-SIM) [10] et non plus uniquement "indistinguable secure" (IND). Actuellement, de nombreux schémas IPE basés sur les hypothèses décisionnelles du résidu composite (Decisional Composite Residuosity (DCR)), du DDH ou du LWE, ont atteint la "SIM-based security" pour un nombre illimité de requêtes. Par la suite, Baltico et al [17] ont construit des schémas de chiffrement fonctionnel quadratique, capables de calculer à partir d'un message x et

d'une matrice fonction A la somme $\sum_{i,j} A_{ij} x_i x_j$. Le premier schéma proposé, efficace et prouvé "selective-secure", repose sur l'hypothèse DDH. Le second schéma, moins efficace mais "adaptive-secure", est basé sur le modèle du groupe générique (Generic Group Model (GGM)). Le premier schéma de chiffrement fonctionnel quadratique efficace basé sur les hypothèses LWE a été proposé par Agrawal, Rosen et al [9].

Dans cette thèse, nous nous concentrerons sur les schémas de FE basés sur le LWE.

1.3 Addendum : chiffrement homomorphe et chiffrement fonctionnel

Le chiffrement homomorphe (HE) est une technique cryptographique permettant la réalisation d'opérations mathématiques sur des données chiffrées au préalable, sans accès à la donnée en clair : seul le propriétaire des données est en capacité de déchiffrer le résultat. L'utilisation du HE dans les protocoles d'apprentissage automatique fait l'objet de recherches actives, mais les progrès dans ce domaine peinent à suivre le rythme de l'augmentation constante des coûts de calcul de l'apprentissage automatique. Par contraste, le chiffrement fonctionnel permet la réalisation d'opérations sur données préalablement chiffrées, avec obtention du résultat en clair : les résultats d'opérations sont obtenus en clair, et seul le propriétaire des données décide des opérations applicables. Souvent considéré comme variante du chiffrement homomorphe, le chiffrement fonctionnel offre cependant un panel sensiblement différent de cas d'applications industriels : tout en garantissant la confidentialité du reste des données brutes, un tiers autorisé est en capacité d'extraire par lui-même une information précise. Curieusement, FE et HE semblent en quelque sorte entrelacés, comme le montrent les travaux de [44, 13]. Une des différences principales entre le FE et le HE est la manière dont les fonctions sont appliquées. Le FE permet de contrôler les fonctions qui peuvent être appliquées aux données par l'intermédiaire d'un détenteur de clé maîtresse, qui émet des clés secrètes en fonction de l'adéquation de la fonction. L'utilisateur utilise ensuite la clé secrète pour obtenir une fonction du texte en clair. Alors que la FHE permet à n'importe qui d'exécuter n'importe quelle fonction avec la clé d'évaluation, le résultat est toujours chiffré. Seul le propriétaire de la clé secrète peut le décrypter. Cependant, les concepts de FHE et FE se chevauchent, et il a été démontré que le chiffrement fonctionnel pouvait fonctionner comme le HE, moyennant une légère adaptation [44].

1.4 Histoire du problème "Learning With Errors"

A la fin du 19 ème siècle, le cryptologue militaire August Kerchkoffs énonce le principe désormais fondamental qui stipule que la sécurité d'un chiffrement doit être assurée même si l'attaquant connaît la technique de chiffrement. Autrement dit, cela implique que la sécurité d'un schéma de chiffrement ne doit pas reposer sur le fait que l'attaquant ignore le schéma. Toute construction en cryptographie est donc basée sur un problème dit difficile, c'est-à-dire un problème mathématique qui ne peut pas être résolu efficacement par un algorithme en un temps polynomial. Pour être en mesure de casser la sécurité d'un schéma de chiffrement, on doit être en mesure de résoudre le problème difficile sur lequel il repose. Les premiers schémas de chiffrement à clé publique sont basés sur des problèmes difficiles de théorie des nombres comme le très populaire schéma RSA [102], dont la sécurité repose sur la complexité de la décomposition en nombres premiers et le problème du logarithme discret dans les groupes cycliques [37].

Cependant, en 1994, le mathématicien Peter Shor [106] met au point un algorithme quantique permettant de résoudre ces deux problèmes. Cet algorithme utilise les propriétés de la mécanique quantique pour effectuer des calculs plus rapidement que les ordinateurs classiques. En particulier, il utilise la transformation de Fourier quantique pour trouver les périodes d'une fonction, ce qui est la clé pour factoriser des nombres entiers. En 2016 l'Institut National de Standard et Technologie (NIST) lance un « process » visant à solliciter, évaluer et standardiser des schémas de chiffrement et des signatures dans un contexte post-quantique. Parmi les pistes initialement retenues pour les schémas de chiffrement, 9 sur 17 sont basés sur des problèmes difficiles liés aux réseaux.

Un réseau euclidien est un sous-groupe discret de l'espace $\mathbb{R}^n$. On peut le voir comme la description d'un maillage régulier ou encore comme l'ensemble des combinaisons linéaires à coefficients entiers de n vecteurs linéairement indépendants. Le problème difficile afférent à ce réseau est le problème dit "du plus court vecteur" (Shortest Vector Problem (SVP)) qui consiste à trouver le plus petit vecteur non nul du réseau au sens de la norme euclidienne.

En cryptographie, les problèmes difficiles utilisés ne sont pas basés sur des réseaux euclidiens standards. En effet, ces derniers présentent des variantes où les problèmes sont trop simples. Ainsi, les réseaux choisis sont ceux pour lesquels les problèmes sont nécessairement difficiles à résoudre. Parmi les problèmes difficiles basés sur les réseaux les plus populaires, on trouve le problème de la solution à entier court (Short Integer Solution (SIS)) présenté par Ajtai [11] et le problème de l'apprentissage avec erreur (Learning

with Error (LWE)) introduit par Regev [100], qui est particulièrement adapté aux schémas de chiffrement. Ce dernier a prouvé que résoudre le problème LWE est en moyenne aussi difficile que de résoudre, avec un algorithme quantique, plusieurs problèmes de réseaux standards dans le pire des cas. Le problème LWE peut être résumé comme suit : il s'agit de retrouver le vecteur secret s à partir de plusieurs échantillons $(a, a \cdot s + e)$ où le masque a est un vecteur tiré uniformément et où e est un bruit gaussien bien choisi. Sa variante décisionnelle (LWE décisionnel) convient particulièrement bien aux constructions cryptographiques et a été créée spécialement pour les schémas de chiffrement à clé publique [100, 50, 79]. Elle est maintenant utilisée dans de nombreux schémas de chiffrement homomorphe [27, 25], et dans certains schémas de chiffrement fonctionnel [3, 8]. De manière générale, les schémas de chiffrement basés sur les problèmes SIS et LWE sont moins efficaces que les protocoles classiques basés sur la théorie des nombres car ils nécessitent le stockage d'une grande matrice A et de nombreux produits matrice-vecteur. Cependant, les variantes SIS, Module-SIS (Module Short Integer Solutions), PLWE (Polynomial Learning With Errors), RLWE (Ring Learning With Errors) et Module-LWE (Module Learning With Errors) sont des variantes de SIS ou de LWE qui utilisent des structures algébriques avantageuses afin d'atteindre une efficacité suffisante en pratique. Dans ces travaux, nous utiliserons des schémas basés sur RLWE [9, 85].

1.5 Contexte des travaux et motivations

Les modèles de Machine Learning (ML) sont basés sur l'apprentissage automatique, c'est-à-dire que les modèles apprennent à partir de données. L'idée est que ces modèles identifient les distributions de probabilités propres à chaque classe en fonction des variables caractérisant les données. Plus il y a de données disponibles, plus les analyses de ces distributions peuvent être affinées par les modèles. Cependant, un petit nombre de données peut ne pas être représentatif des données réelles que l'IA doit classifier. Par conséquent, la quantité et la représentativité des données sont essentielles. De nos jours, l'accès aux données dans le domaine naval de défense est extrêmement difficile, voire impossible. Les modèles d'IA sont souvent créés à partir de bases de données publiques qui représentent au mieux les cas d'utilisation traités. Bien que des techniques telles que l'apprentissage frugal et diverses méthodes d'augmentation de données soient utilisées, elles ont leurs limites. Avoir accès à des bases de données réelles et représentatives en grand nombre permettrait sans aucun doute de proposer des modèles d'IA plus performants. Le FE est

une méthode cryptographique qui permet aux utilisateurs d'accéder de manière sécurisée à une fonction des données sensibles autorisée par le propriétaire des données. Il existe plusieurs options pour le protocole, mais le plus simple est le suivant : le propriétaire des données sensibles chiffre ses données pour les protéger. Le concepteur d'IA demande au propriétaire des données l'accès à certaines fonctions de données. Si le propriétaire juge cela acceptable, il autorise l'accès à la fonction des données en générant et partageant une clé fonctionnelle à l'utilisateur. L'hypothèse sur laquelle repose cette approche est que le propriétaire des données est en mesure de déterminer si les fonctions requises par le concepteur d'IA comportent un risque de compromission pour les données. En pratique, cela dépend du cas d'usage : dans certains cas, les informations sensibles sont facilement identifiables, tandis que dans d'autres cas, il est plus difficile d'évaluer la part de données sensibles. Cependant, il existe des approches IA, telles que les méthodes de visualisation et d'explicabilité ou l'entraînement adversarial, qui aident le propriétaire à évaluer plus précisément quelles informations sont divulguées.

Les avancées récentes dans le domaine de la sécurité des données ont permis le développement de schémas de chiffrement fonctionnel qui permettent d'effectuer des opérations sur des données chiffrées sans avoir à les déchiffrer préalablement. Dans le contexte de la conception de modèles d'IA, cela ouvre de nouvelles perspectives en terme de protection de la confidentialité des données sensibles utilisées dans le processus d'apprentissage . En effet, tout en préservant l'intégrité et la précision des résultats obtenus, cette approche novatrice ouvre la voie au développement de systèmes de machine learning sécurisés et fiables dans le domaine de la défense navale. Cette thèse vise donc à étudier comment ces schémas de chiffrement fonctionnel peuvent être appliqués de manière efficace et sécurisée dans des cas d'application concrets, en particulier dans le domaine de la conception de modèles d'IA en général.

1.6 Nos contributions

1.6.1 État de l'art "Privacy Preserved Machine learning"

Cette thèse s'ouvre sur la réalisation d'un état de l'art approfondi des approches de Privacy-Preserving Machine Learning (PPML). Avec l'augmentation exponentielle de la collecte et de l'utilisation de données sensibles, la protection de la vie privée est devenue une préoccupation majeure dans le domaine de l'apprentissage automatique. Les

approches PPML offrent des solutions prometteuses pour préserver la confidentialité des données tout en permettant des analyses et des modèles prédictifs précis. Cette thèse examine les différentes approches PPML existantes, en évaluant leurs avantages, leurs limites et leur applicabilité dans différents contextes, en fournissant un état de l'art aussi complet que possible des approches PPML.

1.6.2 Chiffrement fonctionnel quadratique rapide pour l'IA

Nous développons des variantes de schémas de chiffrement fonctionnel, linéaires ou quadratiques, basées sur les problèmes DDH et LWE dans des cas concrets de conception de modèles d'intelligence artificielle dont certains cas d'usage sont propres au naval de défense. Pour cela, nous proposons différentes techniques permettant de combiner les modèles d'IA pour répondre aux contraintes du chiffrement fonctionnel. En particulier, nous exploitons la représentation point-valeur des polynômes par la transformée de Fourier ainsi que des réductions de dimension judicieusement choisies.

1.6.3 Méthodologie pour le machine learning basé sur le chiffrement fonctionnel

Nous avons développé une méthodologie novatrice pour adapter la conception d'un modèle de Machine Learning en utilisant le chiffrement fonctionnel. Plus précisément, nous utilisons des techniques basées sur le transfert d'apprentissage et cette approche nous permet d'optimiser les performances du modèle tout en préservant la confidentialité des données sensibles.

1.6.4 Publication pour la conférence CAID 2023

Après un bref historique du chiffrement fonctionnel et quelques définitions utiles, nous proposons une application à un cas d'usage du domaine du naval de défense. En nous appuyant sur un scénario de reconnaissance acoustique sous-marine, nous démontrons ainsi la maturité du FE face à des données réelles en amont d'une conception de modèle par apprentissage automatique profond et en étudions les divers impacts. Enfin, nous discutons de la pertinence du FE et des progrès techniques envisageables à court-terme.

1.6.5 Autres travaux à publier

1. Un schéma de chiffrement fonctionnel quadratique basé sur RLWE combinant ceux de [9] et [85]. Le premier a été utilisé pour mettre en œuvre un chiffrement fonctionnel linéaire, le second nous a permis d'utiliser la fonctionnalité quadratique. Le schéma est présenté dans les Sections 3 et 4.

2. L'application d'un schéma de chiffrement fonctionnel quadratique basé sur RLWE à un modèle d'apprentissage automatique. Nous proposons dans ce travail une adaptation d'un schéma de chiffrement fonctionnel quadratique qui peut être utilisé à la place d'une couche de convolution à fonction d'activation quadratique. En particulier, il permet d'utiliser des filtres de convolution préalablement entraînés comme clés fonctionnelles pour des données privées qui n'ont jamais été vues. L'étape de pré-traitement est expliquée dans la Section 8.1 et une application concrète est détaillée dans la section 8.2.

   Nous proposons aussi un protocole générique détaillé pour utiliser le chiffrement fonctionnel quadratique appliqué à un modèle d'apprentissage automatique. Inspirés par les techniques d'apprentissage par transfert, nous élaborons un protocole de conception permettant l'utilisation sécurisée de modèles pré-entraînés basés sur la convolution et leur ré-entraînement sur des caractéristiques spécifiquement autorisées. Les hypothèses et le protocole sont présentés dans la partie II de la thèse.

1.7 Notations et définitions préliminaires

1.7.1 Notations

Par commodité, nous regroupons dans cette section quelques notations utilisées dans cette thèse. Pour tout entier p non nul, $\mathbb{Z}_p$ désignera l'espace quotient $\mathbb{Z}/p\mathbb{Z}$. Les éléments de $\mathbb{Z}_p^k$ seront les vecteurs à composantes dans $\mathbb{Z}_p$ et les éléments de $\mathbb{Z}_p^{m\times n}$ seront les matrices de taille $m \times n$ à coefficients dans $\mathbb{Z}_p$. Tout au long de cette thèse, nous identifierons les vecteurs avec les matrices lignes, c'est-à-dire que nous identifierons les espaces $\mathbb{Z}_p^k$ et $\mathbb{Z}_p^{1\times k}$.

Sur $\mathbb{R}^k$, nous utiliserons les normes vectorielles usuelles

$$\|x\|_\infty = \max_i\{|x_i|\}, \qquad \|x\|_r = \left(\sum_i |x_i|^r\right)^{1/r} \quad \text{pour } r \geq 1.$$

Sur l'espace des matrices $\mathbb{R}^{m\times n}$, nous aurons besoin de la norme

$$\|A\|_2 = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{i,j})^2\right)^{1/2}.$$

Pour $x \in \mathbb{Z}_p^k$, on peut aussi définir ces normes, étant donné que, pour chaque composante $x_i \in \mathbb{Z}_p$, la quantité $|x_i|$ est définie comme le minimum des valeurs absolues des représentants de x_i. Autrement dit $|x_i|$ est la valeur absolue du *représentant symétrique* de x_i, c'est-à-dire celui qui appartient à $[-p/2, p/2[$. Ce représentant symétrique sera noté $[x]_p$.

Pour tout nombre réel x, on notera $\lfloor x \rfloor$ sa partie entière et $\lfloor x \rceil$ l'entier le plus proche de x (lors que x est demi-entier, on arrondit vers 0 : vers le bas pour les positifs et vers le haut pour les négatifs).

Enfin, nous utiliserons plusieurs types de lois de probabilité. Tout d'abord, pour tout ensemble fini E, la loi de distribution uniforme sur E sera notée $\mathcal{U}(E)$. Dans nos systèmes de chiffrement, le message clair sera caché à l'aide de bruits Gaussiens centrés, qui seront quantifiés par un écart-type.

Définition 1.7.1 (Distributions Gaussiennes)**.** *Soit $k \geq 1$ et $\sigma \in \mathbb{R}_+^*$. Pour tout $x, c \in \mathbb{R}^k$, on note $\rho_{\sigma,c}(x) = \exp(-\|x-c\|_2^2/2\sigma^2)$ la fonction Gaussienne centrée en c et d'écart-type σ. Soit M un sous-groupe additif discret de $\mathbb{R}^k$. On définit sur M la distribution Gaussienne discrète $\mathcal{D}_{M,\sigma,c}$ comme la loi de probabilité ayant la densité*

$$\mathcal{D}_{M,\sigma,c}(x) = \rho_{\sigma,c}(x)/\rho_{\sigma,c}(M), \quad \textit{avec} \quad \rho_{\sigma,c}(M) = \sum_{y\in M}\rho_{\sigma,c}(y).$$

Lorsqu'on a $c = 0$, on omet c dans les notations ci-dessus, qui deviennent ρ_σ et $\mathcal{D}_{M,\sigma}$.

1.7.2 Généralités autour de la cryptographie

Une règle suivie par tous les schémas de chiffrement dont il est question ici, énoncée par le linguiste néerlandais Auguste Kerckhoffs, stipule que la sécurité d'un chiffrement ne devrait reposer que sur le secret de la clé et non sur le secret du chiffrement. Cela peut sembler évident aujourd'hui, alors que les chiffrements et les protocoles sont spécifiés publiquement et utilisés par tout le monde. Mais historiquement, A. Kerckhoffs faisait référence aux machines de chiffrement militaires spécialement conçues pour une armée ou une division donnée. Voici la citation exacte extraite de son essai de 1883 intitulé "La Cryptographie Militaire", où il énumérait six exigences d'un système de chiffrement

militaire : "Le système ne doit pas nécessiter de secret et peut être volé par l'ennemi sans causer de problème."

Définition 1.7.2 (Chiffrement Symétrique)**.** *Un schéma de chiffrement symétrique est un système de chiffrement pour lequel la même clé est utilisée pour chiffrer et déchiffrer les données. Ainsi, si K et M désignent respectivement la clé de chiffrement et le message original, le message chiffré C est obtenu en appliquant une fonction de chiffrement E à la clé K et au message M :*

$$C = E(K, M).$$

De même, si C désigne un message chiffré, alors le message original M dont il est issu est obtenu en appliquant une fonction de déchiffrement D à la clé K et au message chiffré C :

$$M = D(K, C).$$

Définition 1.7.3 (Chiffrement Asymétrique)**.** *Un chiffrement asymétrique est un système de chiffrement qui utilise deux clés différentes pour le chiffrement et le déchiffrement des données. Ces deux clés sont mathématiquement liées, mais il est impossible de déterminer la clé privée* Msk *à partir de la clé publique* Mpk *en un temps polynomial.*

Le chiffrement asymétrique fonctionne de la manière suivante. Soit Mpk *la clé publique et M le message à chiffrer. On obtient le texte chiffré C en appliquant une fonction de chiffrement E à la clé* Mpk *et au message M*

$$C = E(\text{Mpk}, M).$$

Pour déchiffrer le message chiffré C et retrouver le message original M, il suffit d'appliquer une fonction de déchiffrement D à la clé secrète Msk *et au message chiffré C :*

$$M = D(\text{Msk}, C).$$

La sécurité d'un schéma de chiffrement en cryptographie est évaluée en fonction de plusieurs critères, telle que la résistance aux attaques par force brute, aux attaques différentielles et aux attaques par canal auxiliaire.

Définition 1.7.4 (Attaque par force brute)**.** *Une attaque par force brute consiste à essayer toutes les clés possibles pour déchiffrer un message chiffré. Une attaque différentielle est une attaque qui exploite les différences entre les paires de messages chiffrés et les paires*

de clés pour récupérer la clé secrète. Une attaque par canal auxiliaire est une attaque qui exploite les fuites d'informations, telles que la consommation de puissance, le temps d'exécution ou les émissions électromagnétiques, pour récupérer la clé secrète.

Pour évaluer la sécurité d'un schéma de chiffrement, les cryptographes utilisent des méthodes d'analyse mathématiques et des simulations informatiques pour déterminer la résistance du schéma.

Définition 1.7.5 (Algorithme en temps polynomial PPT)**.** *Un algorithme est dit en temps polynomial s'il existe $k, C > 0$ tel que son temps d'exécution sur une entrée de taille n soit au plus Cn^k. De manière équivalente, un algorithme est polynomial s'il existe un $k > 0$, tel que son temps d'exécution sur des entrées de taille n soit est en $O(n^k)$. On dit d'un algorithme de décision calculable en temps polynomial (PPT - Probabilistic Polynomial-Time en anglais) s'il parvient avec une probabilité significativement supérieure à celle d'un oracle aléatoire à résoudre le dit problème.*

Définition 1.7.6 (Problème difficile)**.** *Un problème difficile est un problème mathématique qui est difficile à résoudre avec les moyens actuels de calcul. Plus précisément, un problème difficile est un problème qui nécessite un temps exponentiel pour être résolu, ou encore, un problème pour lequel il n'existe à ce jour aucun algorithme de résolution exécutable en temps polynomial avec les ressources informatiques actuelles.*

1.7.3 Systèmes de chiffrement fondamentaux utilisés dans cette thèse

Dans cette section, nous enchaînons un certain nombre de définitions de systèmes de chiffrement fondamentaux utilisés dans cette thèse, en particulier le problème Learning With Errors (LWE) ainsi que le problème Diffie-Hellman (DH). Nous définirons aussi en fin de section la notion de chiffrement fonctionnel.

Définition 1.7.7 (Problème LWE)**.** *Soient les entiers $n, m \geq 1, p \geq 2$, un paramètre réel $\alpha \in [0, 1]$ et un vecteur secret $s \in \mathbb{Z}_p^n$. Pour $a \in \mathbb{Z}_p^n$ un vecteur tiré uniformément et $e \hookleftarrow \mathcal{D}_{\mathbb{Z}_p, \alpha p}$ un bruit gaussien, on définit la distribution $\mathcal{A}_{n,p,\alpha,s} \in \mathbb{Z}_p^{n+1}$ comme le couple*

$$(a,\ a \cdot s + e \mod p).$$

Le problème $LWE_{n,m,\alpha,p,s}$ standard, aussi appelé problème de recherche LWE, consiste pour l'attaquant à retrouver s à partir de m tirages de $\mathcal{A}_{n,p,\alpha,s}$. Sa variante décisionnelle,

appelée problème de décision LWE consiste, pour l'attaquant, à distinguer les distributions

$$(\mathcal{A}_{n,p,\alpha,s})^m \quad et \quad \mathcal{U}(\mathbb{Z}^{n+1})^m.$$

Définition 1.7.8 (Problème mheLWE)**.** *Soient les entiers $n, m \geq 1, p \geq 2$, un paramètre réel $\alpha \in [0,1]$ et un vecteur secret $s \in \mathbb{Z}_p^n$. Pour $A \in \mathbb{Z}_p^{m \times n}$ une matrice tirée uniformément, $e \hookleftarrow \mathcal{D}_{\mathbb{Z}_p^m, \alpha p}$ un vecteur de bruits gaussiens et $\mathbf{Z} \in \mathbb{Z}_p^{\ell \times m}$ une matrice tirée selon une distribution τ bien choisie [8], on définit la distribution $\mathcal{A}_{n,m,\ell,p,\alpha,\tau,s}$ comme le tuple*

$$(\mathbf{A},\ \mathbf{A} \cdot s + e \mod p,\ \mathbf{Z},\ \mathbf{Z} \cdot e).$$

Le problème LWE multi-indice étendu (Multi-hint extended LWE, noté mheLWE) décisionnel consiste pour l'attaquant à distinguer cette distribution de la distribution uniforme.

Définition 1.7.9 (Problème RLWE)**.** *Le problème Ring Learning With Errors (RLWE) est une extension du problème LWE à un anneau de polynômes quotienté par un polynôme cyclotomique. Soit*

$$R_q = \mathbb{Z}_q[X]/(X^n + 1),$$

où n est une puissance de 2 et q un entier plus grand ou égal à 2. Connaissant un couple $(a, as + e)$ avec a tiré uniformément dans R_q, $s \in R_q$ et e un bruit gaussien bien choisi [81], il n'y a pas d'algorithme PPT capable de retrouver s avec un avantage significatif. On notera $RLWE_{q,n,\sigma}$ ce problème avec paramètres q, n et σ l'écart-type du bruit.

Pour des raisons qui apparaîtront ultérieurement, on prendra p un nombre premier tel que $p \equiv 1 \mod 2n$.

Définition 1.7.10 (Problème du logarithme discret)**.** *Soit G un groupe fini d'ordre n, et soit α un élément de G. Le problème du logarithme discret pour G est le suivant : étant donné un élément $\beta \in G$, trouver un entier x, $0 \leq x \leq n-1$, tel que $\alpha^x = \beta$, si un tel entier existe (c'est-à-dire si β est dans le sous-groupe multiplicatif de G généré par α).*

Définition 1.7.11 (Problème DDH)**.** *Soit un groupe cyclique $\mathbb{G}$ d'ordre premier p et a, b et $c \in \mathbb{Z}_p$ tirés aléatoirement uniformément. On dénote $\mathcal{A}_0$ la distribution du quadruplet*

$$(g, g^a, g^b, g^{ab})$$

et $\mathcal{A}_1$ *la distribution du quadruplet*

$$(g, g^a, g^b, g^c).$$

Le problème décisionnel de Diffie-Hellman (DDH) consiste à distinguer les distributions $\mathcal{A}_0$ *et* $\mathcal{A}_1$*.*

Définition 1.7.12 (Problème DCR)**.** *Soit* $N = pq$ *avec* p *et* q *des nombres premiers. Le problème de la DCR (Decision Composite Residuosity) consiste à distinguer les distributions* $D_0 = \{z = z_0^N \mod N^2 | z_0 \hookleftarrow \mathbb{Z}_{N^2}^*\}$ *et* $D_1 = \{z \hookleftarrow \mathbb{Z}_{N^2}^*\}$ *.*

Définition 1.7.13 (Chiffrement fonctionnel)**.** *Soit* X *un espace de messages,* Y *un espace de messages dual,* E *un espace de résultats et* F *une application définie sur* $X \times Y$ *à valeurs dans* E*. Un schéma de chiffrement fonctionnel pour* F *est un quadruplet d'algorithmes PPT (SETUP, ENCRYPT, KEYGEN, DECRYPT) définis comme suit :*

- *SETUP*$(1^\lambda) \to (\text{Mpk}, \text{Msk})$ *prend en entrée le paramètre de sécurité* λ *et renvoie la clé publique* Mpk *et la clé secrète* Msk*.*
- *ENCRYPT*$(\text{Mpk}, x) \to c_x$ *prend en entrée la clé publique* Mpk *et un message* $x \in X$ *et renvoie un chiffré* c_x *du message.*
- *KEYGEN*$(\text{Msk}, y) \to Sk_y$ *prend en entrée la clé secrète* Msk *et un message dual* $y \in Y$ *et renvoie la clé fonctionnelle* Sk_y *associée.*
- *DECRYPT*$(Sk_y, c_x) \to F(x, y)$ *prend en entrée la clé fonctionnelle* Sk_y *et le chiffré* c_x *et renvoie le résultat* $F(x, y) \in E$*.*

Dans le cas d'un schéma dit IPFE, la fonctionnalité $F(x, y)$ *calculée lors du déchiffrement est le produit scalaire* $x \cdot y$*. Dans ce type de schéma, les espaces* X *et* Y *sont définis tels que* $X = \{0, ..., B_x - 1\}^\ell$ *et* $Y = \{0, ..., B_y - 1\}^\ell$*. Les paramètres* $B_x, B_y \in \mathbb{N}^*$ *sont les nombres de valeurs possibles pour chaque composante, respectivement des vecteurs de messages* x *et des vecteurs de messages duaux* y*. La valeur* $\ell \in \mathbb{N}^*$ *est la taille de ces vecteurs. Dans le cas des schémas IPFE basés sur le RLWE, il est possible de paralléliser le calcul de* n *chiffrements fonctionnels,* n *étant défini par l'anneau polynomial utilisé.*

1.8 Remarques sur la cryptographie post-quantique

Les avancées sur les ordinateurs quantiques représentent une menace potentielle pour les schémas de chiffrement actuels, en particulier ceux basés sur des problèmes tels que RSA ou le logarithme discret. La physique quantique permet de manipuler et de calculer

avec des ensembles de particules subatomiques, appelées qubits, qui peuvent exister simultanément dans plusieurs états. Cela ouvre la porte à des algorithmes de calcul quantique tels que l'algorithme de Shor, qui est capable de factoriser rapidement de grands nombres en utilisant la propriété de superposition quantique. La factorisation de grands nombres est la base de nombreux schémas de chiffrement, y compris RSA. Si un ordinateur quantique suffisamment puissant était développé, il pourrait potentiellement casser ces schémas de chiffrement en un temps beaucoup plus court que les ordinateurs classiques. Le *National Institute of Standards and Technology* (NIST) a donc lancé un processus de sélection pour identifier les algorithmes de chiffrement post-quantiques les plus prometteurs, dans le but de garantir la sécurité des systèmes de chiffrement face aux futures avancées de l'informatique quantique. Le processus de sélection a retenu majoritairement des schémas de chiffrement basés sur les réseaux, en particulier le LWE, qui sont considérés comme sécurisés post-quantiques. Le NIST joue un rôle essentiel dans l'évaluation et la normalisation de ces schémas de chiffrement.

Première partie

Autour des algorithmes de chiffrement fonctionnel

Dans cette thèse, nous avons choisi de nous pencher sur une approche PPML basée sur le chiffrement fonctionnel. Après l'étude de nombreux papiers de recherche, il nous est apparu que les schémas FE basés sur LWE et proposés notamment par [8] étaient très largement reconnus dans le domaine. L'une des valeurs ajoutées de cet article est, de notre point de vue applicatif, l'augmentation du niveau de sécurité des schémas proposés, avec une démonstration de résistance face aux attaques adaptatives. Notons par ailleurs que la plupart des schémas de [8] sont inspirés de ceux de [3] qui ont été démontrés résistants face aux attaques sélectives.

Un niveau de sécurité résistant aux attaques sélectives et adaptatives fait référence à la capacité d'un système cryptographique à prévenir les attaques ciblées, où un attaquant peut choisir les messages qu'il souhaite chiffrer ou déchiffrer, ainsi que les clés utilisées. Plus précisément, une attaque sélective permet à l'attaquant de choisir les messages chiffrés qu'il souhaite déchiffrer, tandis qu'une attaque adaptative lui permet en plus d'ajuster ses choix en fonction des informations qu'il obtient au fur et à mesure de l'attaque.

Les premières constructions de schémas FE, en particulier dans [8], étaient de type IPFE, c'est-à-dire qu'elles permettaient de calculer de manière sécurisée le produit scalaire $x \cdot y$ d'un vecteur de données x avec un vecteur fonctionnel y. Les applications permises par ce type de schémas, bien que ne proposant que du déchiffrement fonctionnel linéaire, sont déjà nombreuses. Nous les avons exploitées pour différents cas d'usage détaillés dans la partie expérimentation, en particulier pour l'ACP, la LDA et les RN et autres CNN linéaires et TF. L'extension à des fonctions non-linéaires (plus précisément quadratiques) a été proposée ensuite dans [9]. Dans un premier temps, nous avons étudié ces schémas avec un regard applicatif, notre objectif étant leur utilisation pour des cas d'usage concrets. Le but était à la fois de comprendre leur fonctionnement et d'en maîtriser les paramètres, afin de pouvoir les adapter à chaque cas d'usage. C'est pourquoi nous avons en particulier étudié les preuves.

Dans cette première partie de la thèse, nous présentons les schémas de chiffrement fonctionnel linéaire et quadratique qui nous ont semblé les plus pertinents pour nos applications, basés sur le chiffrement RLWE. En guise d'introduction, nous décortiquons la version vectorielle (LWE) de ce schéma, qui n'est pas le schéma retenu finalement pour des raisons de coûts de calculs.

Dans la dernière section de cette première partie, nous esquissons enfin plusieurs autres candidats au FE que nous avons aussi étudiés, mais pas retenus. Ils sont quant à eux basés sur le DDH et sur le DCR pour son extension. In fine, ces deux schémas reposent sur le

problème du logarithme discret et appartiennent à la même famille que ceux reposant sur les courbes elliptiques. Ce type de schémas est très utilisé dans la littérature pour les cas d'application à l'IA notamment et il nous semble que cette prédominance s'appuie sur la possibilité de pré-calculer les tables de logarithmes discrets, diminuant ainsi les temps de calcul et améliorant l'efficacité.

Chapitre 2

Schéma de chiffrement fonctionnel linéaire basé sur LWE

2.1 Définition du schéma

Le schéma proposé est basé sur le problème LWE. C'est un schéma IPFE permettant de calculer le produit scalaire $x \cdot y$ entre un vecteur message x de longueur ℓ et de composantes bornées par un entier P et un vecteur fonction y de même longueur et de composantes bornées par V. De la même manière que pour le schéma présenté dans la Section 5.1, le produit scalaire peut être calculé efficacement, uniquement si on le restreint à un petit intervalle, en l'occurrence de taille $K = 2\ell PV$. Pour les calculs ci-dessous, on rappelle que les vecteurs sont identifiés à des matrices lignes, donc le produit scalaire s'écrit indifféremment $x \cdot y = xy^T = yx^T$.

Parameters. *Soient des entiers non nuls p, P, V et un réel $\alpha \in]0, 1[$.*

Set up. *Soient λ le paramètre de sécurité et des paramètres m, n, ℓ (choisis de façon à respecter cette sécurité), tels que $\ell \leq m/2$, on réalise les opérations suivantes :*

1. *Tirage uniforme de la matrice $A \in \mathbb{Z}_p^{m \times n}$ de taille $m \times n$.*
2. *Tirage de la matrice $Z \hookleftarrow \tau^{\ell \times m}$ de taille $\ell \times m$ (la distribution τ est détaillée plus bas).*
3. *Calcul de la matrice $U = ZA \in \mathbb{Z}_p^{\ell \times n}$ de taille $\ell \times n$.*
4. *Calcul de $K = 2\ell PV$. On vérifie que $K \leq p$.*
5. *Définition de la clé secrète maîtresse* $\mathrm{Msk} = Z$ *et de la clé publique maîtresse* $\mathrm{Mpk} = (A, U, K, P, V)$.
6. *Sortie de $LWE.Setup(\lambda, m, n, \ell) = (\mathrm{Msk}, \mathrm{Mpk})$.*

Encryption. *Soient (x, Mpk), avec $x \in \mathbb{Z}_p^\ell$ et $\|x\|_\infty < P$:*

1. *Tirage uniforme de* $s \in \mathbb{Z}_p^n$.
2. *Tirage de* $e_0 \hookleftarrow \mathcal{D}_{\mathbb{Z}_p^m,\sigma}$ *et de* $e_1 \hookleftarrow \mathcal{D}_{\mathbb{Z}_p^\ell,\sigma}$ *suivant des distributions Gaussiennes discrètes avec* $\sigma = \alpha p$.
3. *Calcul de* $c_0 = sA^T + e_0 \in \mathbb{Z}_p^m$ *et* $c_1 = sU^T + \left\lfloor \frac{p}{K} \right\rfloor x + e_1 \in \mathbb{Z}_p^\ell$.
4. *Définition de* $C = (c_0, c_1)$.
5. *Sortie de* $LWE.Encryption\,(x, \mathrm{Mpk}) = C$.

Functional Key generation. *Étant donné* (y, Msk), *avec* $y \in \mathbb{Z}_p^\ell$ *et* $\|y\|_\infty < V$ *:*

1. *Calcul de la clé fonctionnelle* $Sk_y = yZ \in \mathbb{Z}_p^m$.
2. *Sortie de* $LWE.KeyGen\,(y, \mathrm{Msk}) = Sk_y$.

Functional Decryption. *Étant donné* (C, y, Sk_y) *:*

1. *Calcul du déchiffrement fonctionnel*

$$C'_y = yc_1^T - Sk_y c_0^T \in \mathbb{Z}_p.$$

2. *Recherche de la valeur* $C_y \in \{-K/2, ..., K/2\}$ *minimisant* $|\lfloor \frac{p}{K} \rfloor C_y - C'_y|$. *En pratique, on calcule* $C_y = \left\lfloor [C'_y]_p / \lfloor \frac{p}{K} \rfloor \right\rceil$ *(rappelons que* $[C'_y]_p$ *désigne le représentant de* C'_y *dans* $[-p/2, p/2[$*).*
3. *Sortie de* $LWE.Decryption\,(C, y, Sk_y) = C_y$, *qui est bien la quantité voulue* $x \cdot y$, *si les conditions d'exactitude sont respectées (voir le Lemme 2.2.1).*

Le choix de la distribution de τ est lié à la sécurité du schéma de chiffrement. Une distribution bien choisie permet d'évaluer la sécurité grâce à la réduction du problème LWE au problème multi-hint LWE (Définition 1.7.8). En pratique, on peut choisir

$$\tau = \left(\mathcal{D}_{\sigma_1}^{\ell \times m/2} | (\mathcal{D}_{\sigma_2,\delta_1}^{m/2 \times 1} | \cdots | \mathcal{D}_{\sigma_2,\delta_\ell}^{m/2 \times 1})^T\right)$$

avec $\delta_i \in \mathbb{Z}^{m/2}$ le i-ème vecteur canonique (on a bien $\ell \leq m/2$), $\sigma_1 = \Theta(\sqrt{n \log m} \max(\sqrt{m}, K))$ et $\sigma_2 = \Theta(n^{7/2} m^{1/2} \max(m, K^2) \log^{5/2} m)$. Les détails concernant le choix de τ et de la sécurité sont décrits dans le papier [8].

2.2 Condition d'exactitude

Dans cette section, nous démontrons le résultat utile suivant, qui nous autorisera une utilisation pratique du chiffrement fonctionnel linéaire basé sur LWE.

Lemme 2.2.1. *L'exactitude du déchiffrement fonctionnel, tel que décrit dans la Section 2.1, est vérifiée si la condition suffisante suivante est satisfaite :*

$$|\epsilon| < \frac{1}{2}\left\lfloor \frac{p}{K} \right\rfloor, \qquad \text{avec } \epsilon = ye_1^T - yZe_0^T \in \mathbb{Z}_p.$$

Démonstration. Posons $\epsilon = ye_1^T - yZe_0^T$. En rappelant que l'on a

$$c_0 = sA^T + e_0, \quad c_1 = sU^T + \left\lfloor \frac{p}{K} \right\rfloor x + e_1, \quad Sk_y = yZ, \quad U = ZA,$$

on calcule directement

$$yc_1^T - Sk_y c_0^T = yUs^T + \left\lfloor \frac{p}{K} \right\rfloor yx^T + ye_1^T - yZAs^T - yZe_0^T = \left\lfloor \frac{p}{K} \right\rfloor x \cdot y + \epsilon.$$

Rappelons que le déchiffrement fonctionnel s'effectue dans $\mathbb{Z}_p$: on a donc

$$C'_y = \left\lfloor \frac{p}{K} \right\rfloor x \cdot y + \epsilon \mod p.$$

On suppose maintenant que la condition $|\epsilon| < \frac{1}{2}\left\lfloor \frac{p}{K} \right\rfloor$ est satisfaite. Comme $\|x\|_\infty \leq P-1$, $\|y\|_\infty \leq V-1$ et $K = 2\ell PV$, on a

$$|x \cdot y| \leq \ell \|x\|_\infty \|y\|_\infty \leq K/2 - 1.$$

On en déduit que

$$\left| \left\lfloor \frac{p}{K} \right\rfloor x \cdot y + \epsilon \right| \leq \left\lfloor \frac{p}{K} \right\rfloor |x \cdot y| + \epsilon \leq \left\lfloor \frac{p}{K} \right\rfloor \frac{K}{2} - \frac{1}{2}\left\lfloor \frac{p}{K} \right\rfloor < \frac{p}{2}$$

et on a donc $[C'_y]_p = \left\lfloor \frac{p}{K} \right\rfloor x \cdot y + \epsilon$. On peut conclure maintenant que l'arrondi effectué lors du déchiffrement se fait correctement, car

$$C_y := \left\lfloor [C'_y]_p / \left\lfloor \frac{p}{K} \right\rfloor \right\rceil = \left\lfloor x \cdot y + \epsilon / \left\lfloor \frac{p}{K} \right\rfloor \right\rceil = x \cdot y + \left\lfloor \epsilon / \left\lfloor \frac{p}{K} \right\rfloor \right\rceil = x \cdot y,$$

en utilisant encore notre hypothèse sur ϵ. □

2.3 Probabilité d'exactitude

On cherche à estimer la probabilité $\mathbb{P}_\epsilon = \mathbb{P}\left(|\epsilon| < \frac{1}{2}\left\lfloor \frac{p}{K} \right\rfloor\right)$. Le but de cette section est d'expliquer quelles règles de calcul seront implémentées dans nos codes afin de fixer nos paramètres en nous assurant un déchiffrement fonctionnel correct, avec grande probabilité, voir la Section 4.6.

Dans un premier temps, il s'agit de déterminer la loi de

$$\epsilon = y \cdot e_1 - Sk_y \cdot e_0,$$

étant donné que e_0 et e_1 suivent des lois Gaussiennes discrètes, respectivement $\mathcal{D}_{\mathbb{Z}_p^m,\alpha p}$ et $\mathcal{D}_{\mathbb{Z}_p^\ell,\alpha p}$, que l'on a $\|y\|_\infty \leq V$, $Sk_y = yZ$ et que Z est une matrice donnée de taille $\ell \times m$.

Une approximation qui est faite usuellement, valide dans les conditions de nos expériences où le bruit est relativement petit et où p est grand, est d'assimiler les Gaussiennes discrètes à des Gaussiennes continues. De façon plus précise, la notion de smoothing parameters, introduite par Micciancio et Regev [86], permet de quantifier cela.

Lemme 2.3.1. *Étant donnés le vecteur $y \in \mathbb{Z}_p^\ell$ et la matrice $Z \in \mathbb{Z}_p^{\ell\times m}$, la variable aléatoire ϵ suit une loi Gaussienne centrée d'écart type*

$$\sigma = \alpha p\sqrt{\|y\|_2^2 + \|yZ\|_2^2}$$

Démonstration. Le résultat est une conséquence immédiate de l'indépendance entre elles de toutes les composantes de e_0 et e_1. En particulier, on notera que $y \cdot e_1$ suit une loi Gaussienne centrée d'écart type $\alpha p\|y\|_2$ et que $Sk_y \cdot e_0$ suit une loi Gaussienne centrée d'écart type $\alpha p\|yZ\|_2$. □

Par application des Lemmes 2.2.1 et 2.3.1, la probabilité d'avoir un déchiffrement correct est supérieure ou égale à

$$\mathbb{P}_\epsilon = \mathbb{P}\left(|\epsilon| < \frac{1}{2}\left\lfloor \frac{p}{K} \right\rfloor\right) = \mathbb{P}\left(\frac{|\epsilon|}{\sigma} < \frac{1}{2\sigma}\left\lfloor \frac{p}{K} \right\rfloor\right) = \operatorname{erf}\left(\frac{1}{2\sqrt{2}\sigma}\left\lfloor \frac{p}{K} \right\rfloor\right),$$

avec $\sigma = \alpha p\sqrt{\|y\|_2^2 + \|yZ\|_2^2}$. Autrement dit, on a

$$\mathbb{P}_\epsilon = \frac{2}{\sqrt{\pi}} \int_0^{\frac{1}{2\sqrt{2}\sigma}\left\lfloor \frac{p}{K} \right\rfloor} e^{-t^2} dt.$$

À titre d'exemple, on a

$$\mathbb{P}_\epsilon \geq 99,5\% \iff \frac{1}{2\sqrt{2}\sigma}\left\lfloor\frac{p}{K}\right\rfloor \geq 2 \iff \sigma \leq \frac{1}{4\sqrt{2}}\left\lfloor\frac{p}{K}\right\rfloor,$$

de sorte que

$$\mathbb{P}_\epsilon \geq 99,5\% \iff \|y\|_2^2 + \|yZ\|_2^2 \leq \left(\frac{1}{4\sqrt{2}\alpha p}\left\lfloor\frac{p}{K}\right\rfloor\right)^2.$$

On a directement $\|y\|_2 \leq \sqrt{\ell}V$ et, par l'inégalité de Cauchy-Schwarz, $\|yZ\|_2 \leq \|y\|_2\|Z\|_2$, donc

$$\|y\|_2^2 + \|yZ\|_2^2 \leq \ell V^2(1 + \|Z\|_2^2).$$

On peut ainsi conclure de ces inégalités qu'une condition suffisante pour que la probabilité d'avoir un déchiffrement correct soit supérieure à $99,5\%$ est

$$1 + \|Z\|_2^2 \leq \frac{1}{\ell V^2}\left(\frac{1}{4\sqrt{2}\alpha p}\left\lfloor\frac{p}{K}\right\rfloor\right)^2.$$

Le possesseur de la clé secrète maîtresse Z peut tester cette condition après avoir tiré Z et, éventuellement, rejeter cette clé et refaire un tirage.

2.4 Deux variantes de ce schéma

2.4.1 Variante (p_1, p_2)

Nous nommons ainsi une variante du schéma précédent (utile pour la Section 4) où le résultat du déchiffrement fonctionnel tient compte de l'effet d'un modulo p_1 sur les messages.

On introduit dès le départ deux entiers non nuls $p_1 \ll p_2$. L'entier p_2 joue le rôle de p dans le schéma précédent, et désormais on prend $x \in \mathbb{Z}_{p_1}^\ell$, $y \in \mathbb{Z}_{p_1}^\ell$ et on n'a plus besoin de l'entier K. Le schéma précédent est remplacé par le suivant.

Parameters. *Soient des entiers non nuls $p_1 \ll p_2$ et un réel $\alpha \in]0,1[$.*

Set up. *Soient λ le paramètre de sécurité et des paramètres m, n, ℓ (choisis de façon à respecter cette sécurité), avec $\ell \leq m/2$, on réalise les opérations suivantes :*

1. *Tirage uniforme de la matrice $A \in \mathbb{Z}_{p_2}^{m\times n}$ de taille $m \times n$.*
2. *Tirage de la matrice $Z \hookleftarrow \tau^{\ell\times m}$ de taille $\ell \times m$.*

3. *Calcul de la matrice* $U = ZA \in \mathbb{Z}_{p_2}^{\ell \times n}$ *de taille* $\ell \times n$.
4. *Définition de la clé secrète maîtresse* $\mathrm{Msk} = Z$ *et de la clé publique maîtresse* $\mathrm{Mpk} = (A, U)$.
5. *Sortie de* $LWE.Setup(\lambda, m, n, \ell) = (\mathrm{Msk}, \mathrm{Mpk})$.

Encryption. *Soient* (x, Mpk), *avec* $x \in \mathbb{Z}_{p_1}^{\ell}$ *:*

1. *Tirage uniforme de* $s \in \mathbb{Z}_{p_2}^{n}$.
2. *Tirage de* $e_0 \hookleftarrow \mathcal{D}_{\mathbb{Z}_{p_2}^{m},\sigma}$ *et de* $e_1 \hookleftarrow \mathcal{D}_{\mathbb{Z}_{p_2}^{\ell},\sigma}$ *suivant des distributions Gaussiennes discrètes avec* $\sigma = \alpha p_2$.
3. *Calcul de* $c_0 = sA^T + e_0 \in \mathbb{Z}_{p_2}^{m}$ *et* $c_1 = sU^T + \left\lfloor \frac{p_2}{p_1} \right\rfloor [x]_{p_1} + e_1 \in \mathbb{Z}_{p_2}^{\ell}$.
4. *Définition de* $C = (c_0, c_1)$.
5. *Sortie de* $LWE.Encryption\,(x, \mathrm{Mpk}) = C$.

Functional Key generation. *Étant donné* (y, Msk), *avec* $y \in \mathbb{Z}_{p_1}^{\ell}$ *:*

1. *Calcul de la clé fonctionnelle* $Sk_y = [y]_{p_1} Z \in \mathbb{Z}_{p_2}^{m}$.
2. *Sortie de* $LWE.KeyGen\,(y, \mathrm{Msk}) = Sk_y$.

Functional Decryption. *Étant donné* (C, y, Sk_y) *:*

1. *Calcul du déchiffrement fonctionnel*

$$C'_y = [y]_{p_1} c_1^T - Sk_y c_0^T \in \mathbb{Z}_{p_2}.$$

2. *Calcul de* $C_y = \left\lfloor [C'_y]_{p_2} / \lfloor \frac{p_2}{p_1} \rfloor \right\rceil$.
3. *Sortie de* $LWE.Decryption\,(C, y, Sk_y) = C_y$, *qui est bien la quantité voulue* $x \cdot y \mod p_1$ *si les conditions d'exactitude sont respectées (voir le Lemme 2.2.1).*

Notons $\Delta = \left\lfloor \frac{p_2}{p_1} \right\rfloor$ et réexaminons le déchiffrement fonctionnel afin de vérifier qu'il se fait correctement. On peut voir en réalité que la condition précédente $|\epsilon| < \Delta/2$ n'est plus tout à fait suffisante pour assurer le résultat souhaité. Sans que cela ne soit trop gênant, on va donc supposer une condition plus stricte.

Lemme 2.4.1. *L'exactitude du déchiffrement fonctionnel du schéma de la Section 2.1 modifié est vérifiée, c'est-à-dire,*

$$\left\lfloor \frac{p_1}{p_2} [C'_y]_{p_2} \right\rceil = x \cdot y \mod p_1,$$

si la condition suffisante est satisfaite :

$$|\epsilon| < \frac{1}{2}\frac{p_2}{p_1} - \ell\frac{p_1^2}{4}, \qquad \textit{avec } \epsilon = [y]_{p_1} e_1^T - [y]_{p_1} Z e_0^T \in \mathbb{Z}_{p_2}.$$

De plus, si on suppose que $p_1/p_2(1+\ell p_1^2) < 1$ *, alors une condition suffisante est simplement* $|\epsilon| \leq \frac{1}{4}\Delta$.

Démonstration. En reprenant la preuve du Lemme 2.2.1, on voit que l'on a

$$[C'_y]_{p_2} = \Delta x \cdot y + \epsilon \ mod\ p_2.$$

Donc il existe un entier k tel que

$$[C'_y]_{p_2} = \frac{p_2}{p_1}[x \cdot y]_{p_1} + (x \cdot y)\delta + \epsilon + kp_2, \tag{2.1}$$

avec $\delta = \lfloor \frac{p_2}{p_1} \rfloor - \frac{p_2}{p_1}$. En multipiliant par p_1/p_2 et en prenant l'arrondi on obtient

$$\left\lfloor \frac{p_1}{p_2}[C'_y]_{p_2} \right\rceil = [x \cdot y]_{p_1} + kp_1 + \left\lfloor \frac{p_1}{p_2}\left((x \cdot y)\delta + \epsilon\right) \right\rceil$$

Rappelons que l'on a pris dès le départ dans ce schéma les représentants $x = [x]_{p_1}$ et $y = [y]_{p_1}$. Cela implique que

$$|x \cdot y| \leq \ell p_1^2/4.$$

Par conséquent, si

$$|\epsilon| < \frac{1}{2}\frac{p_2}{p_1} - \ell\frac{p_1^2}{4},$$

alors

$$\left|\frac{p_1}{p_2}\left((x \cdot y)\delta + \epsilon\right)\right| \leq \frac{p_1}{p_2}\left(\ell\frac{p_1^2}{4} + |\epsilon|\right) < \frac{1}{2}.$$

Sous cette condition sur ϵ on a alors

$$\left\lfloor \frac{p_1}{p_2}[C'_y]_{p_2} \right\rceil = [x \cdot y]_{p_1} + kp_1 = [x \cdot y]_{p_1} \ mod\ p_1.$$

On peut maintenant vérifier facilement que si $p_1/p_2(1+\ell p_1^2) < 1$ alors $\frac{1}{4}\Delta < \frac{1}{2}\frac{p_2}{p_1} - \ell\frac{p_1^2}{4}$, et la condition $|\epsilon| \leq \frac{1}{4}\Delta$ suffit donc pour assurer l'exactitude. Ceci conclut la preuve du lemme. □

2.4.2 Variante de déchiffrement par prise de modulo

Nous esquissons ici une autre variante du schéma de la Section 2, où le message n'est plus placé sur les bits de poids forts dans le chiffré (cela s'opérait grâce à la multiplication par $\lfloor p/K \rfloor$), mais sur les bits de poids faible. La méthode de déchiffrement diffère et se fait, non plus par un arrondi, mais par une prise de modulo. On se donne deux entiers non nuls $p_2 \geq p_1$, l'espace des chiffrés est $\mathbb{Z}_{p_2}$ tandis que l'espace des messages est $\mathbb{Z}_{p_1}$.

Parameters. *Soient des entiers non nuls p_1, p_2 tels que $p_2 \gg p_1$ et soit un réel $\alpha \in]0,1[$.*

Set up. *Soient λ le paramètre de sécurité et des paramètres m, n, ℓ (choisis de façon à respecter cette sécurité), avec $\ell \leq m/2$, on réalise les opérations suivantes :*

1. *Tirage uniforme de la matrice $A \in \mathbb{Z}_{p_2}^{m \times n}$ de taille $m \times n$.*
2. *Tirage de la matrice $Z \in \mathbb{Z}_{p_2}^{\ell \times m}$ de taille $\ell \times m$, selon une distribution τ comme ci-dessus.*
3. *Calcul de la matrice $U = ZA \in \mathbb{Z}_{p_2}^{\ell \times n}$ de taille $\ell \times n$.*
4. *Définition de la clé secrète maîtresse* $\mathrm{Msk} = Z$ *et la clé publique maîtresse* $\mathrm{Mpk} = (A, U)$.
5. *Sortie de $LWE.Setup(\lambda, m, n, \ell) = (\mathrm{Msk}, \mathrm{Mpk})$.*

Encryption. *Soient (x, Mpk), avec $x \in \mathbb{Z}_{p_1}^{\ell}$:*

1. *Tirage uniforme de $s \in \mathbb{Z}_{p_2}^{n}$.*
2. *Tirage de $e_0 \hookleftarrow \mathcal{D}_{\mathbb{Z}_{p_2}^m, \sigma}$ et de $e_1 \hookleftarrow \mathcal{D}_{\mathbb{Z}_{p_2}^{\ell}, \sigma}$ suivant des distributions Gaussiennes discrètes avec $\sigma = \alpha p_2$.*
3. *Calcul de $c_0 = sA^T + p_1 e_0 \in \mathbb{Z}_{p_2}^m$ et $c_1 = sU^T + [x]_{p_1} + p_1 e_1 \in \mathbb{Z}_{p_2}^{\ell}$.*
4. *Définition de $C = (c_0, c_1)$.*
5. *Sortie de $LWE.Encryption\,(x, \mathrm{Mpk}) = C$.*

Functional Key generation. *Étant donné (y, Msk), avec $y \in \mathbb{Z}_{p_1}^{\ell}$:*

1. *Calcul de la clé fonctionnelle $Sk_y = [y]_{p_1} Z \in \mathbb{Z}_{p_2}^m$.*
2. *Sortie de $LWE.KeyGen\,(y, \mathrm{Msk}) = Sk_y$.*

Functional Decryption. *Étant donné (C, y, Sk_y) :*

1. *Calcul du déchiffrement fonctionnel*

$$C_y = \left([y]_{p_1} c_1^T - Sk_y c_0^T \mod p_2 \right) \mod p_1.$$

2. *Sortie de* $LWE.Decryption\,(C, y, Sk_y) = C_y$, *qui est bien la quantité voulue* $x \cdot y \in \mathbb{Z}_{p_1}$, *si les bruits* e_0 *et* e_1 *sont suffisamment petits et si* $p_1 \ll p_2$.

Chapitre 3

Chiffrement fonctionnel linéaire basé sur RLWE

Dans ce chapitre, nous nous intéressons désormais à un schéma de chiffrement fonctionnel linéaire basé sur RLWE. Il semble que les schémas basés sur le LWE, bien que résistants post-quantiques, ont fait l'objet de moins de travaux que ceux basés sur DDH ou sur d'autres extensions du problème du logarithme discret. Notre revue de littérature nous a amené à considérer le problème RLWE, extension polynomiale du LWE (voir la Définition 1.7.9), comme un problème difficile particulièrement approprié pour construire des schémas de chiffrement fonctionnels efficaces. L'exploitation du problème RLWE offre un avantage en termes de temps de calcul par rapport aux schémas basés sur LWE ou sur multi-hint LWE (Définition 1.7.8).

La clé de voûte de notre construction est donc un schéma fonctionnel linéaire basé sur RLWE. Nous avons tout d'abord construit un tel schéma en généralisant les schémas LWE de la littérature. Entretemps, un schéma similaire a été publié [85], répondant aux mêmes exigences mais, surtout, proposant une preuve approfondie de sécurité. C'est une adaptation de ce schéma que nous avons choisi de présenter ici. Cette adaptation inclut la possibilité de paralléliser les opérations de chiffrement et de déchiffrement et sera, dans un second temps, combinée avec le schéma fonctionnel quadratique proposé par [9] (voir Section 4).

3.1 Définition du schéma

En rappelant la notation

$$R_p = \mathbb{Z}_p[X]/(X^n + 1),$$

n étant une puissance de 2 (le polynôme $X^n + 1$ est le 2n-ième polynôme cyclotomique), nous présentons dans cette section une version du schéma précédent de chiffrement linéaire

fonctionnel dans ces anneaux de polynômes. Ce schéma RLWE est sécurisé face aux attaques adaptatives et a été proposé (sous une version légèrement différente) par [85]. Il permet le calcul du produit scalaire $x \cdot y$ entre un vecteur de messages polynomiaux $x \in R_p^\ell$ et un vecteur de fonctions polynomiales $y \in R_p^\ell$, avec des coefficients pas trop grands (voir plus loin les bornes précisées). La sécurité du schéma repose sur le problème RLWE défini dans la partie 1.7.9.

Comme pour les schémas précédents, le schéma proposé suit quatre étapes : (Set up) qui décrit la génération des clés maîtres publique et secrète Msk, de la Mpk et des différents paramètres, (Encryption) qui est l'étape de passage de l'espace des messages à l'espace des chiffrés, (Key generation) qui consiste à générer des clés fonctionnelles et (Decryption) qui calcule le produit scalaire dans l'espace des messages à partir de l'espace des chiffrés.

Notations complémentaires. Dans tout ce qui suit, la notation R_p^ℓ désigne un vecteur (ou, selon notre convention, une matrice ligne) dont les ℓ composantes sont des éléments de R_p. De même, $R_p^{\ell \times m}$ désigne une matrice dont les $\ell \times m$ coefficients sont dans R_p.

Si $x \in R_p$ et $r \in [1, \infty]$ sa norme $\|x\|_r$ désigne la norme r du vecteur constitué par les coefficients de ce polynôme (dans la pratique, on n'utilisera que $r = 2$ et $r = \infty$). De plus, si $x = (x_1, \cdots, x_\ell) \in R_p^\ell$, on pose

$$\|x\|_\infty = \max_i \|x_i\|_\infty, \qquad \|x\|_2 = \left(\sum_{i=1}^{\ell} \|x_i\|_2^2 \right)^{1/2}.$$

Enfin, par prolongement de la notation déjà introduite, on désigne par $x \hookrightarrow \mathcal{D}_\sigma$ un polynôme $x \in R_p$ dont tous les coefficients sont tirés selon une distribution Gaussienne discrète centrée d'écart-type σ.

Parameters. *Soient des entiers non nuls p, B_x, B_y et des réels $\alpha_1, \alpha_2, \alpha_3 \in]0, 1[$.*

Set up. *Soit λ le paramètre de sécurité et soient des paramètres ℓ, m, n (choisis de façon à respecter cette sécurité), on réalise les opérations suivantes :*

1. *Tirage de la clé secrète maîtresse* $\mathrm{Msk} = Z \hookleftarrow \mathcal{D}_{\alpha_1 p}^{\ell \times m}$.
2. *Tirage uniforme de $a \in R_p^m$.*
3. *Calcul de $U = aZ^T \in R_p^\ell$.*
4. *Calcul de $K = 2n\ell B_x B_y$. On vérifie que $K \leq p$.*
5. *Définition de la clé publique maîtresse* $\mathrm{Mpk} = (a, U, K, B_x, B_y)$.
6. *Sortie de $LinFE.Setup(\lambda, \ell, m) = (\mathrm{Msk}, \mathrm{Mpk})$.*

Encryption. *Étant donné* (x, Mpk)*, avec* $x \in R_p^\ell$ *et* $\|x\|_\infty < B_x$ *:*

1. *Tirage de* $s \hookleftarrow \mathcal{D}_{\alpha_2 p}$ *à partir d'une distribution Gaussienne.*
2. *Tirages des bruits Gaussiens* $e_0 \hookleftarrow \mathcal{D}_{\alpha_2 p}^m$ *et* $e_1 \hookleftarrow \mathcal{D}_{\alpha_3 p}^\ell$*.*
3. *Calculs de*
$$b_0 = sa + e_0 \in R_p^m,$$
$$b_1 = sU + \left\lfloor \frac{p}{K} \right\rfloor x + e_1 \in R_p^\ell$$
4. *Définition du chiffré* $b = (b_0, b_1)$*.*
5. *Sortie de* $LinFE.Encryption\,(x, \mathrm{Mpk}) = b$*.*

Functional Key generation. *Étant donné* (y, Msk)*, avec* $y \in R_p^\ell$ *et* $\|y\|_\infty < B_y$ *:*

1. *Calcul de la clé fonctionnelle* $Sk_y = yZ \in R_p^m$*.*
2. *Sortie de* $LinFE.KeyGen\,(y, \mathrm{Msk}) = Sk_y$*.*

Functional Decryption. *Étant donné* (b, y, Sk_y) *:*

1. *Calcul du déchiffrement fonctionnel*
$$r = y \cdot b_1 - b_0 \cdot Sk_y$$
2. *Sortie de* $LinFE.Decryption\,(b, y, Sk_y) = \left\lfloor [r]_p / \left\lfloor \frac{p}{K} \right\rfloor \right\rceil$*, qui est bien* $x \cdot y$ *si les conditions d'exactitude sont respectées (voir le Lemme 3.2.1).*

3.2 Conditions d'exactitude

Nous présentons dans cette partie les conditions d'exactitude du déchiffrement fonctionnel linéaire de notre schéma : l'exactitude est assurée si le bruit global du schéma, noté $\epsilon^{(1)}$ ne dépasse pas une certaine borne explicite.

Lemme 3.2.1. *L'exactitude du déchiffrement fonctionnel linéaire, tel que décrit dans la Section 3.1, est vérifiée si la condition suffisante est satisfaite :*

$$\left\|\epsilon^{(1)}\right\|_\infty < \frac{1}{2}\left\lfloor \frac{p}{K} \right\rfloor \qquad \text{avec } \epsilon^{(1)} = ye_1^T - yZe_0^T \in R_p.$$

Démonstration. En posant $\epsilon^{(1)} = ye_1^T - yZe_0^T$, on obtient (rappelons que $U = aZ^T$ et que s est un simple polynôme) :

$$\begin{aligned} r &= yb_1^T - Sk_y b_0^T \\ &= syU^T + \left\lfloor \frac{p}{K} \right\rfloor yx^T + ye_1^T - syZa^T - yZe_0^T \\ &= \left\lfloor \frac{p}{K} \right\rfloor x \cdot y + \epsilon^{(1)}. \end{aligned}$$

Par ailleurs, comme tous les polynômes constituant les vecteurs $x \in R_p^\ell$ et $y \in R_p^\ell$ satisfont respectivement $\|x_i\|_\infty < B_x$ et $\|y_i\|_\infty < B_y$, il est facile de montrer, en utilisant les règles de multiplication modulo $X^n + 1$, que

$$\|x \cdot y\|_\infty < n\ell B_x B_y = K/2.$$

La fin de la démonstration est maintenant la même que celle du Lemme 2.2.1. Si la condition $\left\|\epsilon^{(1)}\right\|_\infty < \frac{1}{2}\left\lfloor \frac{p}{K} \right\rfloor$ est satisfaite, on aura bien $\left\lfloor [r]_p / \left\lfloor \frac{p}{K} \right\rfloor \right\rceil = x \cdot y$. □

3.3 Probabilité d'exactitude

Nous présentons dans cette partie une méthode pour estimer la probabilité d'exactitude du déchiffrement fonctionnel linéaire RLWE décrit ci-dessus. Pour cela, nous évaluons la loi de probabilité du bruit $\epsilon^{(1)}$ du schéma et nous nous appuyons sur le Lemme 3.2.1 pour le calcul de la borne supérieure du bruit.

D'après le Lemme 3.2.1, nous savons que l'exactitude est vérifiée si

$$\left\|\epsilon^{(1)}\right\|_\infty < \frac{1}{2}\left\lfloor \frac{p}{K} \right\rfloor,$$

avec

$$\epsilon^{(1)} = ye_1^T - yZe_0^T \in R_p.$$

Calculons la loi de $\epsilon^{(1)}$. Pour cela, rappelons que les distributions des coefficients des composantes de ϵ_0 sont tirés selon $\mathcal{D}_{\alpha_2 p}$ et celles de e_1 sont selon $\mathcal{D}_{\alpha_3 p}$ (ils sont tous indépendants deux à deux). Les variables Z et y sont considérées ici comme fixées. Nous en déduisons qu'un coefficient $\epsilon_i^{(1)}$ de $\epsilon^{(1)}$, avec $i \in \{0, ..., n-1\}$, est une somme de variables

Gaussiennes indépendantes, et suit une distribution Gaussienne $\mathcal{D}_{\sigma_{\epsilon^{(1)}}}$, d'écart-type

$$\sigma_{\epsilon^{(1)}} = \sqrt{\sigma_2^2 + \sigma_3^2},$$

avec

$$\sigma_2 = \alpha_2 p \|yZ\|_2 \quad \text{et} \quad \sigma_3 = \alpha_3 p \|y\|_2.$$

Nous souhaiterions enfin en déduire la probabilité d'exactitude

$$\mathbb{P}^{(1)} = \mathbb{P}\left(\left\|\epsilon^{(1)}\right\|_\infty < \frac{1}{2}\left\lfloor\frac{p}{K}\right\rfloor\right).$$

En toute rigueur, il est très difficile d'évaluer exactement cette probabilité, même si l'on sait que chaque coefficient suit une loi normale avec variance connue. En effet, ces coefficients ne sont pas indépendants entre eux. Heureusement, plusieurs travaux se sont attachés à étudier ce qui se passe lorsque la taille des polynômes n est grande. On pourra par exemple citer [34], où il est démontré, en utilisant le théorème central limite, que dans la pratique tout se passe comme si les coefficients étaient indépendants. On peut donc, avec confiance, s'attendre à pouvoir utiliser l'expression

$$\mathbb{P}^{(1)} = \operatorname{erf}\left(\frac{1}{2\sqrt{2}\sigma_{\epsilon^{(1)}}}\left\lfloor\frac{p}{K}\right\rfloor\right)^n.$$

Nous avons mené des expérimentations qui ont montré que cette heuristique est tout à fait valide pour les valeurs de n supérieures à 1000 utilisées ici.

3.4 Estimation de la sécurité

L'estimation de la sécurité de ce schéma est basée sur l'utilisation couplée des résultats de [85] [1] et du *Lattice Estimator* [12]. Le *Lattice Estimator* est un outil collaboratif mis en ligne par l'équipe de Martin R. Albrecht (King's College, Londres), outil régulièrement actualisé, qui permet notamment, à partir des principales attaques connues, d'évaluer la sécurité pratique des schémas LWE et RLWE.

En suivant [85], on peut montrer que notre schéma se comporte comme un schéma RLWE ayant pour paramètres n, p et σ, où σ joue le rôle d'un "écart-type équivalent" et

1. Nous avons pu bénéficier de l'aide d'Angshuman Karmakar, l'un des auteurs de [85], via un script qu'il nous a fourni, permettant de calculer cet écart-type équivalent σ et la sécurité du schéma.

est calculé de façon à satisfaire les trois conditions suivantes :

$$\sigma_1 = \alpha_1 p = 2B_x\sqrt{\ell} + 1,$$

$$\sigma_2 = \alpha_2 p = \sigma n\sqrt{\ell\kappa}\sigma_1,$$

$$\sigma_3 = \alpha_3 p = 2\sigma n\sqrt{\kappa}\sigma_1.$$

3.5 Choix des paramètres

Nous proposons une méthode de sélection de paramètres du schéma garantissant une probabilité d'exactitude, dans notre cas $\mathbb{P}^{(1)} \geq 0.995$, et un niveau de sécurité S, le tout dans un objectif de minimisation du temps de calcul.

En pratique, les paramètres B_x, B_y et ℓ sont imposés par le cas d'application visé. Le paramètre n, correspond au nombre de calculs effectués en parallèle lors d'un chiffrement fonctionnel, est également choisie en fonction du cas d'usage. Notons que n à un impact significatif sur la sécurité du schéma. En nous inspirants de la publication [85], nous choisissons de prendre à chaque fois deux jeux paramètres avec

1. $n = 1024$
2. $n = 2048$

Les autres paramètres seront à déterminer. La méthode proposée repose sur un processus itératif qui vise à trouver un point de fonctionnement en ajustant progressivement les paramètres afin d'atteindre le niveau d"exactitude $\mathbb{P}^{(1)}$ et de sécurité S.

Pour assurer la contrainte de conception $p \gg K$, on initialise le paramètre p à une valeur de l'ordre de $10^2 \times K$. Le processus itératif se décompose selon les étapes suivantes :

1. A l'aide du *Lattice Estimator* [12], on choisit un écart type σ permettant d'atteindre le niveau de sécurité S souhaité.
2. On détermine ensuite les paramètres $\alpha_1, \alpha_2, \alpha_3$, correspondants à l'écart type σ identifié en utilisant l'évaluation de sécurité décrite partie 3.4.
3. On évalue la probabilité d'exactitude $\mathbb{P}^{(1)}$ comme décrit dans la partie 3.3.
4. Si on a $\mathbb{P}^{(1)} < 0.995$, alors on augmente p jusqu'à obtenir $\mathbb{P}^{(1)} \geq 0.995$.
5. On effectue une évaluation de la sécurité comme expliqué dans la partie 3.4.
6. Si le niveau de sécurité obtenu est inférieur à S, on réitère le processus. Sinon, le processus se termine et on conserve les paramètres.

3.6 Variante (p_1, p_2) de ce schéma RLWE

Similairement à la Section 2.4.1, nous présentons une variante du schéma précédent (avec p renommé p_2) où l'on prend exactement un modulo p_1 sur le message déchiffré. On a maintenant $x \in R_{p_1}^{\ell}$ et $y \in R_{p_1}^{\ell}$ et on remplace le schéma précédent par celui-ci :

Parameters. *Soient deux entiers $p_1 \ll p_2$ et des réels $\alpha_1, \alpha_2, \alpha_3 \in]0, 1[$.*

Set up. *Soit λ le paramètre de sécurité et soient des paramètres ℓ, m (choisis de façon à respecter cette sécurité), on réalise les opérations suivantes :*

1. *Tirage de la clé secrète maîtresse* $\mathrm{Msk} = Z \hookleftarrow \mathcal{D}_{\alpha_1 p_2}^{\ell \times m}$.
2. *Tirage uniforme de* $a \in R_{p_2}^{m}$.
3. *Calcul de* $U = aZ^T \in R_{p_2}^{\ell}$.
4. *Définition de la clé publique maîtresse* $\mathrm{Mpk} = (a, U)$.
5. *Sortie de* $LinFE.Setup(\lambda, \ell, m) = (\mathrm{Msk}, \mathrm{Mpk})$.

Encryption. *Étant donné* (x, Mpk), *avec* $x \in R_{p_1}^{\ell}$:

1. *Tirage de* $s \hookleftarrow \mathcal{D}_{\alpha_2 p_2}$ *à partir d'une distribution Gaussienne.*
2. *Tirages du bruit Gaussien* $e_0 \hookleftarrow \mathcal{D}_{\alpha_2 p_2}^{m}$ *et du bruit Gaussien* $e_1 \hookleftarrow \mathcal{D}_{\alpha_3 p_2}^{\ell}$.
3. *Calcul de*
$$b_0 = sa + e_0 \in R_{p_2}^{m},$$
$$b_1 = sU + \left\lfloor \frac{p_2}{p_1} \right\rfloor [x]_{p_1} + e_1 \in R_{p_2}^{\ell}$$
4. *Définition du chiffré* $b = (b_0, b_1)$.
5. *Sortie de* $LinFE.Encryption\,(x, \mathrm{Mpk}) = b$.

Functional Key generation. *Étant donné* (y, Msk), *avec* $y \in R_{p_2}^{\ell}$:

1. *Calcul de la clé fonctionnelle* $Sk_y = [y]_{p_1} Z \in R_{p_2}^{m}$.
2. *Sortie de* $LinFE.KeyGen\,(y, \mathrm{Msk}) = Sk_y$.

Functional Decryption. *Étant donné* (b, y, Sk_y) :

1. *Calcul du déchiffrement fonctionnel*
$$r = [y]_{p_1} \cdot b_1 - b_0 \cdot Sk_y$$

2. *Sortie de* $LinFE.Decryption\,(b, y, Sk_y) = \left\lfloor [r]_{p_2} / \left\lfloor \frac{p_2}{p_1} \right\rfloor \right\rceil$, *qui est bien* $x \cdot y \mod p_1$ *si les conditions d'exactitude sont respectées (voir le Lemme 3.6.1).*

En combinant et adaptant les preuves des Lemmes 3.2.1 et 2.4.1, on obtient le résultat suivant.

Lemme 3.6.1. *On suppose le rapport* p_1/p_2 *assez petit. Alors, l'exactitude du déchiffrement fonctionnel linéaire, tel que décrit ci-dessus, est vérifiée si la condition suffisante suivante est satisfaite :*

$$\left\|\epsilon^{(1)}\right\|_\infty \leq \frac{1}{4}\left\lfloor \frac{p_2}{p_1} \right\rfloor \qquad \text{avec } \epsilon^{(1)} = ye_1^T - yZe_0^T \in R_{p_2}.$$

Chapitre 4

Chiffrement fonctionnel quadratique basé sur RLWE

Au cours de notre première salve d'expérimentations décrites dans la Section 7, nous avons constaté que les transformations réalisables par IPFE, c'est-à-dire les transformations linéaires, imposent une contrainte sur l'architecture des modèles choisis.

En effet, nous avons intégré le FE linéaire dans la chaîne fonctionnelle d'un modèle de deux manières. Soit en tant que pré-traitement linéaire comme l'ACP (voir Section 7.1.2), la LDA (voir Section 7.2.2) ou la Transformée de Fourier dans le cas de la reconnaissance acoustique sous marine (voir Section 7.4.2), soit en intégrant le chiffrement fonctionnel directement à l'intérieur d'un modèle de réseau de neurones afin de sécuriser le calcul d'une couche du réseau (voir Section 7.3.2). Cependant, les réseaux de neurones ont besoin de fonctions d'activation non linéaires pour apprendre des relations complexes entre les variables d'entrée et de sortie. Il est possible d'intégrer de la non linéarité dans les modèles en utilisant des fonctions d'activation quadratiques (voir Section 8.2.2) (au sens de la mise au carré), et d'obtenir ainsi de meilleures performances de modèles, voir partie 8.2.

Des schémas de chiffrement permettant de calculer une fonction quadratique des données par chiffrement fonctionnel existent [17, 121, 7, 9] . On distingue deux catégories de schémas quadratiques : les schémas basés sur les courbes elliptiques [17, 121, 7] via les couplages, et les schémas basés sur les réseaux Euclidiens [9].

Nous avons également remarqué que des travaux ont été rapportés sur l'application de schémas de chiffrement fonctionnel quadratiques basés sur le chiffrement de type Diffie-Hellman à des cas d'utilisation de l'apprentissage automatique [38]. Cependant, à notre connaissance, bien qu'il existe des schémas de chiffrement fonctionnel quadratiques basés sur le LWE [9], aucune application spécifique à l'apprentissage automatique n'a encore été proposée. Notons de plus que les problèmes de réseaux en général et le problème RLWE (voir Section 1.7.9) en particulier sont considérés comme une base solide pour la construction de chiffrements post-quantiques, comme expliqué dans le paragraphe 1.8.

Pour implémenter du chiffrement fonctionnel quadratique dans des modèles d'IA, nous nous sommes appuyés sur le schéma proposé par [9] qui utilise une technique permettant de calculer une fonction quadratique en utilisant des propriétés homomorphes des chiffrés, spécifique au chiffrement fonctionnel (FE).

A partir d'un chiffrement LWE standard, il est possible de calculer une fonction du chiffrement en utilisant une sur-couche de chiffrement fonctionnelle linéaire. La technique utilisée est spécifiquement détaillée en partie 4.1.

Le schéma de chiffrement fonctionnel quadratique proposé par [9] est polynômial et utilise de façon sous-jacente, sans le décrire explicitement, le schéma de chiffrement fonctionnel linéaire vectoriel (LWE) de [8]. Nous proposons une version du même schéma mais utilisant le schéma de chiffrement fonctionnel linéaire polynômial décrit dans la Section 3. Nous avons privilégié le format polynômial par rapport au format vectoriel car les schémas basés sur le RLWE sont plus efficaces que les schémas vectoriels basés sur le LWE ou le multi-hint LWE, grâce à la structure algébrique des polynômes permettant de chiffrer naturellement plusieurs messages dans un seul chiffré RLWE et ainsi d'effectuer des calculs en parallèle [9].

Nous proposons également une méthode afin d'évaluer la sécurité de l'ensemble du schéma et trouver les paramètres adaptés selon le cas d'usage traité. En particulier, nous proposons une nouvelle approche pour évaluer la sécurité du déchiffrement linéaire (voir Section 3.4).

4.1 Présentation synthétique de la méthode

La méthode permettant de calculer une fonction quadratique du message d'entrée repose sur un chiffrement en cascade proposé par [9]. Plus précisément, le message d'entrée θ (un vecteur de polynômes) est chiffré une première fois, puis un sur-chiffrement fonctionnel linéaire est appliqué à une fonction bien choisie du chiffré c obtenu. Le déchiffrement fonctionnel linéaire permet ainsi d'obtenir une fonction du premier chiffré nécessaire à l'obtention de la fonction quadratique voulue après un deuxième déchiffrement. Les paragraphes ci-après détaillent formellement ce processus, qui peut sembler un peu complexe à première vue.

On se donne deux entiers non nuls $p_1 \ll p_2$. L'objectif est de permettre l'accès sécurisé à une fonction quadratique $\sum_{1 \leq j \leq i \leq w} g_{ij}\theta_i\theta_j$ d'un message $\theta \in R^w$ dont les composantes sont

bornées en norme infinie par un certain B. Ce message est chiffré par RLWE selon

$$c_i = u_i s^{(1)} + \left\lfloor \frac{p_1}{K_1} \right\rfloor \theta_i + e_i^{(1)} \in R_{p_1}, \quad \text{pour} \quad i \in \{1, ..., w\},$$

avec $u \in R_{p_1}^w$, $s^{(1)} \in R_{p_1}$ une clé secrète tirée uniformément, $e_i^{(1)} \hookleftarrow \mathcal{D}_\sigma$ un bruit Gaussien, σ étant bien choisi (voir les Sections 4.2 et 4.5) et K_1 étant une constante définie dans la Section 4.2. Comme noté par [9], un calcul direct donne

$$\left\lfloor \frac{p_1}{K_1} \right\rfloor^2 \theta_i \theta_j = c_i c_j + u_i u_j (s^{(1)})^2 - u_i c_j s^{(1)} - u_j c_i s^{(1)} + E_{i,j} \mod p_1$$

avec

$$E_{i,j} = -\left\lfloor \frac{p_1}{K_1} \right\rfloor \theta_j e_i^{(1)} - \left\lfloor \frac{p_1}{K_1} \right\rfloor \theta_i e_j^{(1)} - e_i^{(1)} e_j^{(1)}.$$

En multipliant par les coefficients g_{ij} de notre fonction-objectif et en sommant sur les indices i et j, nous obtenons

$$\begin{aligned} \left\lfloor \frac{p_1}{K_1} \right\rfloor^2 \sum_{1 \le j \le i \le w} g_{ij} \theta_i \theta_j &= \sum_{1 \le j \le i \le w} g_{ij} \left(c_i c_j + u_i u_j s^{(1)^2} - u_i c_j s^{(1)} - u_j c_i s^{(1)} \right) + E \mod p_1 \\ &= \sum_{1 \le j \le i \le w} g_{ij} c_i c_j + \Omega + E \mod p_1, \end{aligned} \tag{4.1}$$

avec

$$\Omega = \sum_{1 \le j \le i \le w} g_{ij} \left(u_i u_j s^{(1)^2} - u_i c_j s^{(1)} - u_j c_i s^{(1)} \right) \qquad \text{et} \qquad E = \sum_{1 \le j \le i \le w} g_{ij} E_{i,j}.$$

Notre objectif est de donner un accès sécurisé à $\left\lfloor \frac{p_1}{K_1} \right\rfloor^2 \sum_{1 \le j \le i \le w} g_{ij} \theta_i \theta_j$ en utilisant cette formule (4.1). On considère cette équation en deux parties. Le premier terme $\sum_{1 \le j \le i \le w} g_{ij} c_i c_j$ est sécurisé et peut être directement partagé. Un accès sécurisé au second terme Ω est quant à lui obtenu à l'aide d'un chiffrement fonctionnel linéaire décrit dans la Section 4.5.

Le chiffrement fonctionnel linéaire (précisément, sa variante (p_1, p_2) comme définie dans la Section 3.6) permet de calculer $x \cdot y \mod p_1$ à partir d'un message (secret) x et d'une fonction (publique) décrite par y, donc en posant

$$\begin{cases} x = \left((s^{(1)})^2, c_1 s^{(1)}, ..., c_w s^{(1)} \right) \in R_{p_1}^{w+1}, \\ y = \sum_{1 \le j \le i \le w} g_{ij} \left(u_i u_j, 0...0, -u_i, 0...0, -u_j, 0...0 \right) \in R_{p_1}^{w+1} \end{cases}$$

(où, au sein du vecteur de taille $w+1$ dans la somme définissant y, $-u_i$ est placé à la place $j+1$ et $-u_j$ est placé à la place $i+1$), nous permettons un accès sécurisé à $x \cdot y = \Omega$. Il reste maintenant à sécuriser Ω lui même. En effet, sachant que c, u et g sont connus, Ω est un polynôme de $s^{(1)}$ et doit être protégé afin d'éviter que l'on puisse retrouver $s^{(1)}$ facilement. Pour cela on introduit un bruit supplémentaire Gaussien $\eta \in R_{p_1}^Q$ et on redéfinit les deux vecteurs concaténés suivants :

$$\begin{cases} \widetilde{x} = (x|\eta) \in R_{p_1}^{w+1+Q}, \\ \widetilde{y} = (y|e_k) \in R_{p_1}^{w+1+Q}, \end{cases}$$

où e_k est le k-ième vecteur canonique de $R_{p_1}^Q$, de sorte que

$$\widetilde{x} \cdot \widetilde{y} = \Omega + \eta_k, \qquad \text{avec } k \text{ choisi dans } \{1, ..., Q\}.$$

L'ajout du bruit η permet donc d'accéder à Ω tout en protégeant $s^{(1)}$, comme dans un chiffré RLWE usuel (voir la Section 4.5), il sera par la suite annulé lors du déchiffrement fonctionnel de la même manière que le bruit $e^{(1)}$ (voir la Section 4.3) . À ce stade, à partir de Ω et des termes précédemment partagés, il est possible de calculer

$$\sum_{1 \leq j \leq i \leq w} g_{ij} c_i c_j + \widetilde{x} \cdot \widetilde{y} = \left\lfloor \frac{p_1}{K_1} \right\rfloor^2 \sum_{1 \leq j \leq i \leq w} g_{ij} \theta_i \theta_j \text{ (+ du bruit)} \mod p_1.$$

Enfin, après calcul de l'arrondi $\left\lfloor \left(\sum_{1 \leq j \leq i \leq w} g_{ij} c_i c_j + \Omega \right) / \left\lfloor \frac{p_1}{K_1} \right\rfloor^2 \right\rceil$, nous obtenons le résultat souhaité, à savoir $\sum_{1 \leq j \leq i \leq w} g_{ij} \theta_i \theta_j$.

Notons que le schéma que nous proposons ici est une variante de la technique de [9] basée sur une annulation du bruit par arrondi plutôt que par prise de modulo (comme pour le schéma LWE de la Section 2.4.2). Ce choix est fait dans l'optique de combiner le schéma de [9] avec celui développé par [85]. De plus, on note également que le schéma est potentiellement vulnérable à trois endroits : chiffrement initial par c de θ, chiffrement fonctionnel linéaire par b de x (ou respectivement c_b dans le schéma introduit dans la Section 4.5) et enfin chiffrement fonctionnel linéaire donnant accès uniquement à $\Omega + \eta_k$. Une méthode permettant d'évaluer la sécurité de l'ensemble est donnée dans la Section 4.5.

L'un des avantages d'un tel système de chiffrement fonctionnel est qu'il permet, grâce à la dimension du vecteur de bruit η, des requêtes multiples – Q dans notre cas – tout

en ne chiffrant les messages qu'une seule fois, toujours sous réserve d'autorisation d'accès aux clés fonctionnelles par le propriétaire de la clé secrète maîtresse Msk : le propriétaire conserve un contrôle total sur l'accès fonctionnel à θ. En capacité de choisir arbitrairement de donner accès ou non à une fonction de θ, il peut, comme dans un schéma à mémoire présenté dans [8] (que nous présentons dans la Section 5.3), se souvenir des fonctions qu'il a autorisées et s'assurer qu'un utilisateur ne soit pas capable de reconstruire les parties non autorisées de θ.

4.2 Définition du schéma

Dans cette partie, nous présentons l'intégralité du schéma de chiffrement fonctionnel quadratique. Le schéma de chiffrement fonctionnel quadratique étudié est une variante de celui publié par [9]. L'annulation du bruit lors du déchiffrement fonctionnel repose sur l'arrondi de la même manière que pour le schéma introduit par [85]. Il permet de calculer une fonction quadratique

$$\sum_{1\leq j\leq i\leq w} g_{ij}\theta_i\theta_j$$

à partir d'un message, qui est un vecteur de polynômes $\theta \in R^w$, et d'un vecteur donné $g \in \mathbb{Z}^{w(w+1)/2}$, où $w(w+1)/2$ est le cardinal de l'ensemble $\{1 \leq j \leq i \leq w\}$. Cette construction s'appuie sur le schéma de chiffrement fonctionnel linéaire RLWE, noté LinFE, qui a été présenté dans la Section 3.6. Nous utilisons deux entiers p_1 et p_2 tels que $p_1 \ll p_2$, où p_1 est la taille de l'espace des chiffrés et p_2 est la taille de l'espace des sur-chiffrés dans le schéma LinFE. Comme pour les schémas précédents, le schéma proposé suit quatre étapes : (Set up) génération des clés maîtres publiques et secrètes Mpk, Msk et des différents paramètres à éventuellement tirer aléatoirement, (Encryption) passage de l'espace des messages à l'espace des chiffrés, (Key generation) génération des clés fonctionnelles et (Decryption) calcul de la fonction quadratique dans l'espace des messages à partir de l'espace des chiffrés.

Parameters. *Soient $B, p_1, p_2 \in \mathbb{N}^*$ des entiers tels que $p_1 \ll p_2$, soit $w \in \mathbb{N}^*$, et des réels $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in]0, 1[$.*

Set up. *Soit λ le paramètre de sécurité et soient des entiers non nuls m et Q (choisis tels qu'*in fine*, la sécurité soit respectée), on réalise les opérations suivantes :*

1. *Génération de la clé secrète maîtresse* Msk *et de la clé publique maîtresse* Mpk *en*

utilisant LinFE.Setup.

$$(\mathrm{Msk}, \mathrm{Mpk}) = LinFE.Setup\,(\lambda, \ell := w + 1 + Q, m)\,,$$

c'est-à-dire $\mathrm{Mpk} = (a, U)$ *avec* $a \in R_{p_2}^m$, $U \in R_{p_2}^{w+1+Q}$ *et* $\mathrm{Msk} = Z$, *avec* $Z \in R_{p_2}^{(w+1+Q)\times m}$.

2. *Tirage uniforme de* $u \in R_{p_1}^w$.
3. *Calcul de*
$$K_1 = 1 + \sqrt{w(w+1)p_1 B^2 \|g\|_\infty}$$
et vérification que $K_1 \leq p_1$.
4. *Output* $= (\mathrm{Msk}, \mathrm{Mpk}, u, B, K_1)$.

Encryption. *Étant donné* $(\theta, \mathrm{Mpk}, u, B, K_1)$, *avec un message* $\theta \in R_{p_1}^w$ *tel que* $\|\theta\|_\infty < B$ *:*

1. *Tirage uniforme de la clé secrète* $s^{(1)} \in R_{p_1}$.
2. *Tirage d'un bruit Gaussien* $e^{(1)} \hookleftarrow \mathcal{D}_{\alpha_4 p_1}^w$.
3. *Calcul du chiffré*
$$c = s^{(1)} u + \left\lfloor \frac{p_1}{K_1} \right\rfloor \theta + e^{(1)} \in R_{p_1}^w.$$
4. *Tirage d'un bruit Gaussien* $\eta \hookleftarrow \mathcal{D}_{\alpha_4 p_1}^Q$.
5. *Calcul de*
$$\widetilde{x} = \left({s^{(1)}}^2, c_1 s^{(1)}, ..., c_w s^{(1)}, \eta_1, ..., \eta_Q\right) \in R_{p_1}^{w+1+Q}.$$
6. *Calcul du chiffré* b *en utilisant le chiffrement linéaire RLWE LinFE.Encryption*
$$b = LinFE.Encryption\,(X, \mathrm{Mpk}) \in R_{p_2}^m \times R_{p_2}^{w+1+Q},$$
dans sa variante (p_1, p_2) *(schéma de la Section 3.6).*
7. *Output* $= (c, b)$.

Functional Key generation. *Étant donnés* $(g, a, \mathrm{Msk}, k, u)$, *avec* $g \in \mathbb{Z}^{w(w+1)/2}$ *:*

1. *Calcul du vecteur polynômial*
$$y = \sum_{1 \leq j \leq i \leq w} g_{ij}\,(u_i u_j, 0...0, -u_i, 0...0, -u_j, 0...0) \in R_{p_1}^{w+1}.$$

avec $-u_i$ positionné dans la $j+1$-ième composante et $-u_j$ dans la $i+1$-ième composante des termes sommés.

2. *Définition de $\widetilde{y} = (y|e_k)$, la concaténation entre y et e_k, le vecteur e_k étant le k-ième vecteur canonique de longueur Q.*
3. *Calcul de la clé fonctionnelle en utilisant la procédure de génération de clé fonctionnelle linéaire LinFE.KeyGen*

$$Sk_g = LinFE.KeyGen\left(\widetilde{y}, \mathrm{Msk}\right) \in R_{p_2}^m.$$

4. *Output $= (\widetilde{y}, Sk_g)$.*

Functional Decryption. *Étant donné $(g, c, b, \widetilde{y}, Sk_g, K_1)$:*

1. *Calcul du déchiffrement fonctionnel quadratique à l'aide de la procédure de déchiffrement fonctionnel linéaire*

$$d = \sum_{1 \leq j \leq i \leq w} g_{ij} c_i c_j + LinFE.Dec\left(b, \widetilde{y}, Sk_g\right) \mod p_1.$$

2. *Sortie de $Decryption(g, c, b, \widetilde{y}, Sk_g) = \left\lfloor [d]_{p_1} / \left\lfloor \frac{p_1}{K_1} \right\rfloor^2 \right\rceil$, qui est bien*

$$\sum_{1 \leq j \leq i \leq w} g_{ij} \theta_i \theta_j$$

si les conditions d'exactitude sont respectées (voir Section 4.3).

4.3 Conditions d'exactitude

Nous allons nous appuyer sur les conditions d'exactitude du déchiffrement fonctionnel linéaire RLWE, qui ont été étudiées en détail dans la Section 3.2.

Lemme 4.3.1. *On fait l'hypothèse que la condition énoncée dans le Lemme 3.6.1 est satisfaite, où e_0 et e_1 sont les bruits générés lors de l'étape de calcul de $b = LinFE.Encryption\left(\widetilde{x}, \mathrm{Mpk}\right)$.*

Alors, l'exactitude du déchiffrement fonctionnel quadratique, tel que décrit dans la Section 4.5, est vérifiée si la condition suivante est satisfaite :

$$\left\| \epsilon^{(2)} \right\|_\infty < \frac{1}{2} \left\lfloor \frac{p_1}{K_1} \right\rfloor^2, \tag{4.2}$$

avec

$$\epsilon^{(2)} = \eta_k + \sum_{1\leq j\leq i\leq w} g_{ij}\left(\left\lfloor\frac{p_1}{K_1}\right\rfloor\theta_i e_j^{(1)} + \left\lfloor\frac{p_1}{K_1}\right\rfloor\theta_j e_i^{(1)} + e_i^{(1)}e_j^{(1)}\right).$$

Démonstration. Rappelons que l'on a posé

$$d = \sum_{1\leq j\leq i\leq w} g_{ij}c_i c_j + LinFE.Dec\left(b, \widetilde{y}, Sk_g\right).$$

En utilisant le Lemme 3.2.1 énonçant que le déchiffrement fonctionnel est exact, on obtient (modulo p_1)

$$\begin{aligned} d &= \sum_{1\leq j\leq i\leq w} g_{ij}c_i c_j + \widetilde{x}\cdot\widetilde{y} \\ &= \sum_{1\leq j\leq i\leq w} g_{ij}\left(c_i c_j + u_i u_j (s^{(1)})^2 - u_i c_j s^{(1)} - u_j c_i s^{(1)}\right) + \eta_k \\ &= \left\lfloor\frac{p_1}{K_1}\right\rfloor^2 \sum_{1\leq j\leq i\leq w} g_{ij}\theta_i\theta_j + \epsilon^{(2)}. \end{aligned}$$

Finalement, en choisissant le représentant de d dans $[-p_1/2, p_1/2]$, on obtient bien

$$Decryption(g, c, b, \widetilde{y}, Sk_g) := \left\lfloor [d]_{p_1} / \left\lfloor\frac{p_1}{K_1}\right\rfloor^2\right\rceil = \sum_{1\leq j\leq i\leq w} g_{ij}\theta_i\theta_j,$$

sous l'hypothèse

$$\left\|\epsilon^{(2)}\right\|_\infty < \frac{1}{2}\left\lfloor\frac{p_1}{K_1}\right\rfloor^2.$$

Il faut noter que la taille de K_1 a été calibrée de telle sorte que l'on soit certain d'avoir

$$\left|\left\lfloor\frac{p_1}{K_1}\right\rfloor^2 \sum_{1\leq j\leq i\leq w} g_{ij}\theta_i\theta_j\right| < \frac{p_1}{2}.$$

□

4.4 Probabilité d'exactitude

Nous présentons dans cette partie un moyen d'estimer la probabilité $\mathbb{P}^{(2)}$ d'exactitude du déchiffrement fonctionnel quadratique (dans un premier temps, en supposant que l'étape de déchiffrement fonctionnel linéaire s'est faite correctement). Pour cela, nous évaluons la loi de probabilité du bruit $\epsilon^{(2)}$ du schéma et nous nous appuyons sur le Lemme

4.3 pour le calcul de la borne supérieure du bruit.

D'après le Lemme 4.3, nous savons que l'exactitude est vérifiée si l'on a

$$\left\|\epsilon^{(2)}\right\|_\infty < \frac{1}{2}\left\lfloor\frac{p_1}{K_1}\right\rfloor^2,$$

avec

$$\epsilon^{(2)} = \eta_k + \sum_{1\leq j\leq i\leq w} g_{ij}\left(\left\lfloor\frac{p_1}{K_1}\right\rfloor e_j^{(1)}\theta_i + \left\lfloor\frac{p_1}{K_1}\right\rfloor e_i^{(1)}\theta_j + e_i^{(1)}e_j^{(1)}\right).$$

Pour déterminer la loi de $\epsilon^{(2)}$, rappelons les distributions $e^{(1)} \hookleftarrow \mathcal{D}^w_{\alpha_4 p_1}$ et $\eta_k \hookleftarrow \mathcal{D}_{\alpha_4 p_1}$. Le vecteur de polynômes θ est fixé ici. Par ailleurs, on va faire une hypothèse simplificatrice (largement vérifiée dans la pratique) : le coefficient α_4 étant supposé petit, tous les termes en erreurs quadratiques $e_i^{(1)}e_j^{(1)}$ sont négligeables.

On peut déduire de l'indépendance de toutes ces variables aléatoires qu'un coefficient $\epsilon_i^{(2)}$ de $\epsilon^{(2)}$ est une somme de variables Gaussiennes indépendantes et suit donc une distribution Gaussienne $\mathcal{D}_{\sigma_{\epsilon^{(2)}}}$ avec

$$\sigma^2_{\epsilon^{(2)}} = \alpha_4^2 p_1^2\left(1 + \left\lfloor\frac{p_1}{K_1}\right\rfloor^2 \|(G+G^T)\theta\|_2^2\right),$$

et où on note G la matrice triangulaire inférieure $G = (g_{ij})_{1\leq j\leq i\leq w}$. Pour évaluer la probabilité d'exactitude

$$\mathbb{P}^{(2)} = \mathbb{P}\left(\left\|\epsilon^{(2)}\right\|_\infty < \frac{1}{2}\left\lfloor\frac{p_1}{K_1}\right\rfloor^2\right),$$

il nous faut procéder comme dans la Section 3.3. En effet, en toute rigueur, cette probabilité est extrêmement difficile à prédire à partir d'uniquement les variances des coefficients de ce polynôme d'erreur. Toutefois, comme n est grand, nous pouvons encore nous appuyer sur le théorème centrale limite, qui autorise à utiliser avec très bonne confiance la formule suivante :

$$\mathbb{P}^{(2)} = \mathrm{erf}\left(\frac{1}{2\sqrt{2}\sigma_{\epsilon^{(2)}}}\left\lfloor\frac{p_1}{K_1}\right\rfloor^2\right)^n.$$

Bien entendu, la probabilité finale $\mathbb{P}$ que le déchiffrement soit exacte doit inclure le calcul de la probabilité $\mathbb{P}^{(1)}$ effectué dans la Section 3.3 (en tenant compte du fait que la condition de déchiffrement correct est la condition plus restrictive donnée dans le Lemme 3.6.1). On note que les variables aléatoires intervenant dans les calculs de $\mathbb{P}^{(1)}$ et $\mathbb{P}^{(2)}$ sont

indépendantes. Donc, on a

$$\mathbb{P} = \mathbb{P}^{(1)} \times \mathbb{P}^{(2)} = \operatorname{erf}\left(\frac{1}{4\sqrt{2}\sigma_{\epsilon^{(1)}}}\left\lfloor\frac{p_2}{p_1}\right\rfloor\right)^n \operatorname{erf}\left(\frac{1}{2\sqrt{2}\sigma_{\epsilon^{(2)}}}\left\lfloor\frac{p_1}{K_1}\right\rfloor^2\right)^n.$$

4.5 Estimation de la sécurité

Dans cette partie nous proposons une méthode permettant d'évaluer la sécurité du schéma fonctionnel quadratique détaillé dans la Section 4.2 et de choisir des paramètres permettant d'atteindre un niveau de sécurité souhaité. On peut distinguer dans ce schéma trois opérations pour lesquelles l'évaluation de la sécurité est nécessaire :

1. La sécurité du chiffrement initial.
2. La sécurité du calcul de la quantité Ω masquée par le bruit η dans l'étape de déchiffrement fonctionnel linéaire. Ce calcul intervient dans le déchiffrement fonctionnel quadratique.
3. La sécurité du chiffrement fonctionnel linéaire utilisé.

La valeur globale de la sécurité du schéma est définie comme le minimum de ces trois valeurs de sécurité à calculer. Notre approche pratique destinée à évaluer quantitativement la sécurité consiste en l'ajustement successif des paramètres à mesure de la prise en compte de ces trois sécurités, dans cet ordre. Cela signifie que nous fixons dans un premier temps les paramètres afférents à la sécurité du chiffrement initial, puis ceux liés au calcul de Ω et enfin les paramètres du chiffrement fonctionnel linéaire. De fait, l'évaluation de la sécurité du chiffrement linéaire utilisé ainsi que le choix des paramètres nécessite d'avoir connaissance de la valeur p_1 qui est dépendante des deux autres évaluations de la sécurité.

L'évaluation de la sécurité pour le chiffrement initial (point 1), c'est à dire la sécurité du chiffrement $c = s^{(1)}u + \left\lfloor\frac{p_1}{K_1}\right\rfloor\theta + e^{(1)} \in R_{p_1}^w$, repose sur la sécurité du problème RLWE (voir Section 1.7.9) avec les paramètres p_1, n et $\sigma = \alpha_4 p_1$.

L'évaluation de la sécurité du chiffrement linéaire (point 3) utilisé ainsi que le choix des paramètres sont expliqués dans les Sections 3.4 et 3.5.

Il reste le point 2, c'est-à-dire l'évaluation de la sécurité du calcul de $\Omega + \eta_k$, avec

$$\Omega = \sum_{1\leq j\leq i\leq w} g_{ij}\left(u_i u_j s^2 - u_i c_j s - u_j c_i s\right).$$

Ce problème de type RLWE n'est pas tout à fait standard et nous avons choisi, pour évaluer sa sécurité, de produire un lemme original de réduction d'un problème RLWE plus

classique avec les paramètres p, n, σ vers ce problème avec p, n et $\sigma_\eta = \sqrt{1 + \log_2(p)\frac{n}{2}}\sigma$. Nous ne prétendons pas que ce résultat est optimal, mais il nous permet de donner une minoration de la sécurité de ce problème. Dans la pratique, l'augmentation de l'écart-type reliée à cette réduction ne sera pas pénalisante.

Définition 4.5.1 (Réduction)**.** *On dit qu'il existe une réduction d'un problème A à un problème B s'il existe un algorithme en temps polynomial permettant de résoudre A étant donné un oracle résolvant B.*

Le principe des réductions est le suivant. Étant donné une instance de A, on transforme cette instance (en temps polynomial) en instance de B pour utiliser l'oracle, puis on transforme la réponse de l'oracle (en temps polynomial) en réponse pour le problème A. C'est équivalent à montrer que si on sait résoudre B alors on sait résoudre A : si A est réputé sûr, cela rend B sûr.

Lemme 4.5.2. *Soit deux entiers non nuls $t < p$, un message $x \in R_t$, une clé $s \in R_p$ et a_0, a_1, a_2 des masques tirés uniformément sur R_p. Soient deux bruits Gaussiens e et e' tirés selon des distributions $\mathcal{D}_\sigma$ et $\mathcal{D}_{\sigma'}$.*

On pose $RLWE_{p,n,\sigma}(x) = (a_0, b)$, le chiffré de x selon

$$b = a_0 s + \left\lfloor \frac{p}{t} \right\rfloor x + e \in R_p.$$

et $RLWE'_{p,n,\sigma'}(x) = (a_1, a_2, b')$, le chiffré de x selon

$$b' = a_1 s^2 + a_2 s + \left\lfloor \frac{p}{t} \right\rfloor x + e' \in R_p.$$

Il existe une réduction de $RLWE_{p,n,\sigma}$ vers $RLWE'_{p,n,\sigma'}$ où

$$\sigma' = \sigma\sqrt{1 + \lceil \log_2(p) \rceil \frac{n}{2}}.$$

Dans ce lemme, $\log_2$ désigne le logarithme en base 2.

Démonstration. On considère une instance $RLWE_{p,n,\sigma}(x) = (a_0, b)$ avec

$$b = a_0 s + \left\lfloor \frac{p}{t} \right\rfloor x + e \in R_p.$$

On veut la transformer en une instance RLWE'. On suppose avoir (sans mettre en cause la confidentialité de s) une donnée supplémentaire : une liste de chiffrés $RLWE_{p,n,\sigma}(2^j s^2)$

qui jouent le rôle d'une clé publique, c'est-à-dire des couples (a_{0j}, b_j) avec $j = 0, \cdots, m = \lceil \log_2(p) \rceil - 1$ et

$$b_j = a_{0j}s + 2^j s^2 + e_{sj} \in R_p,$$

sachant que $a_{0j} \hookleftarrow \mathcal{U}(R_p)$ et $e_{sj} \hookleftarrow \mathcal{D}_\sigma$.

Soit u tiré uniformément sur R_p. On décompose u en base 2 en posant $u = \sum_{j=0}^{m} u_j 2^j$ où $u_j \in R_2$. On pose ensuite

$$\begin{cases} a_1 = u, \\ a_2 = a_0 + \sum_{j=0}^{m} u_j a_{0j}, \\ b' = b + \sum_{j=0}^{m} u_j b_j. \end{cases}$$

Par construction a_1 est tiré uniformément sur R_p et le fait que a_0 soit uniforme sur R_p entraîne que a_2 est aussi uniforme sur R_p. Maintenant, on calcule

$$\begin{aligned} b' - a_1 s^2 - a_2 s &= b + \sum_{j=0}^{m} u_j b_j - u s^2 - a_0 s - \sum_{j=0}^{m} u_j a_{0j} s \\ &= b - a_0 s - u s^2 + \sum_{j=0}^{m} u_j (b_j - a_{0j} s) \\ &= b - a_0 s - u s^2 + \sum_{j=0}^{m} u_j \left(2^j s^2 + e_{sj}\right) \\ &= b - a_0 s + \sum_{j=0}^{m} u_j e_{sj} \\ &= \left\lfloor \frac{p}{t} \right\rfloor x + e + \sum_{j=0}^{m} u_j e_{sj}. \end{aligned}$$

On obtient bien une instance $(a_1, a_2, b') = RLWE'(x)$ avec un bruit Gaussien $e' = e + \sum_{j=0}^{m} u_j e_{sj} \hookrightarrow \mathcal{D}_{\sigma'}$, où

$$\sigma' = \sigma \sqrt{1 + \frac{n(m+1)}{2}}.$$

□

4.6 Choix des paramètres

Dans cette section, nous proposons une méthode de sélection des paramètres du schéma pour atteindre un niveau de sécurité spécifié S. Nous proposons ensuite, à partir de ces derniers, une méthode de sélection de paramètres garantissant une probabilité d'exactitude, dans notre cas $\mathbb{P}^{(2)} \geq 0.995$, et le niveau de sécurité S, le tout dans un objectif de minimisation du temps de calcul.

En pratique, les paramètres p_0, g et w sont imposés par l'application visée. Le paramètre n, correspondant au nombre de calculs équivalents effectués en parallèle lors d'un chiffrement fonctionnel, est également choisi en fonction du cas d'usage. Notons que la valeur de n a un impact significatif sur la sécurité du schéma. En nous inspirant de la publication [85], nous choisissons de prendre deux jeux de paramètres avec

1. $n = 1024$;
2. $n = 2048$;

Les autres paramètres à déterminer sont d'une part $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ qui mesurent les dimensions des bruits pour les différents schémas, et d'autre part p_1 et p_2, qui mesurent respectivement la taille de l'espace du chiffrement initial et celle de l'espace du sur-chiffrement comme expliqué dans la Section 4.4.

Nous recherchons ces paramètres à l'aide d'un processus itératif qui vise à trouver un point de fonctionnement en ajustant progressivement les paramètres afin d'atteindre le niveau d'exactitude $\mathbb{P}^{(2)}$ et de sécurité S.

On suppose nos vecteurs de données θ (de taille w) bornées par un certain B. Pour assurer la contrainte de conception $K_1 \leq p_1 < p_2$, avec $K_1 = 1 + \sqrt{w(w+1)p_1 B^2 \|g\|_\infty}$, on initialise les paramètres p_1 et p_2 à des nombres Sections de l'ordre de respectivement $10^2 \times K_1$ et $10^4 \times K_1$. Le processus itératif se décompose selon les étapes suivantes :

1. A l'aide du "Lattice Estimator" [12], on identifie empiriquement un écart type σ permettant d'atteindre le niveau de sécurité S souhaité.
2. On détermine ensuite les paramètres $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ correspondants à l'écart type σ identifié en utilisant les méthodes d'évaluation de sécurité décrites dans les parties 3.4 et 4.5.
3. On évalue la probabilité d'exactitude $\mathbb{P}^{(2)}$ comme décrit dans la partie 4.3.
4. Si $\mathbb{P}^{(2)} < 0.995$, alors on augmente, par ordre de priorité, p_1 puis p_2 jusqu'à obtenir $\mathbb{P}^{(2)} \geq 0.995$

5. On effectue une évaluation de la sécurité comme expliqué dans les Sections 3.4 et 4.5.
6. Si le niveau de sécurité obtenu est inférieur à S, on réitère le processus. Sinon, le processus se termine et on conserve les paramètres.

Je tiens à exprimer ma gratitude envers Angshuman Karmakar, l'auteur de la publication [85] pour m'avoir conseillé le processus itératif proposé.

Notons que p_1 correspond à la fois à la taille de l'espace des chiffrés dans le chiffrement initial et à la taille des messages d'entrée du sur-chiffrement fonctionnel linéaire (voir Section 3.1) . Le paramètre p_2 correspond à la taille de l'espace des chiffrés du sur-chiffrement fonctionnel linéaire (voir Section 3.1). Lors de l'étape 4 du processus itératif, il est donc recommandé de fixer en premier p_1 car il impacte l'exactitude du déchiffrement linéaire. On augmente donc dans l'ordre p_1 puis p_2 pour obtenir la probabilité d'exactitude $\mathbb{P}^{(2)}$ souhaitée. Notons que plus p_1 et p_2 sont élevés, plus le temps de calcul augmente. On choisit donc les valeurs de p_1 et p_2 minimales permettant d'obtenir l'exactitude.

Chapitre 5

Présentation de quelques autres candidats au FE

Dans ce chapitre, afin de rendre notre présentation plus complète, nous présentons quelques autres schémas de FE que nous n'avons pas retenu pour nos expérimentations.

5.1 Chiffrement Fonctionnel IPFE basé sur l'hypothèse Décisionnelle de Diffie-Hellman (DDH)

Nous nous intéressons ici à un schéma de chiffrement fonctionnel, proposé par [8], basé sur l'hypothèse décisionnelle de Diffie Hellman (voir Section 1.7.10), réductible au problème du logarithme discret (voir Section 1.7.11). Ce schéma est une adaptation de celui proposé par [3]. Il en conserve notamment la même efficacité. La construction proposée atteint un niveau de sécurité résistant aux attaques adaptatives.

Définition du schéma

Le schéma suivant est un schéma de chiffrement IPFE permettant de calculer le produit scalaire $x \cdot y$ d'un vecteur message $x \in \mathbb{Z}_p^\ell$ chiffré avec un vecteur fonction $y \in \mathbb{Z}_p^\ell$ choisi, avec ℓ un entier correspondant à la taille de message et p un nombre premier à définir dans la phase de mise en place du schéma.

Set up. *Étant donné $\lambda \in \mathbb{N}^*$ le paramètre de sécurité et $\ell \in \mathbb{N}^*$ la taille du vecteur message $x \in \mathbb{Z}_p^\ell$:*

1. *On choisit un groupe cyclique $\mathbb{G}$ d'ordre p premier, avec $p < 2^\lambda$, et deux générateurs associés $g, h \hookleftarrow \mathbb{G}$.*
2. *On tire uniformément $s, t \in \mathbb{Z}_p$.*
3. *On calcule $\forall i \in \{1, ..., \ell\}, h_i = g^{s_i} \times h^{t_i}$.*

4. *On définit la clé secrète maîtresse* $\text{Msk} = (s, t)$ *et la clé publique maîtresse* $\text{Mpk} = \left(\mathbb{G}, g, h, \{h_i\}_{i=1}^{\ell}\right)$.
5. *On renvoie* $DDH.Setup(\lambda, \ell) = (\text{Msk}, \text{Mpk})$.

Encryption. *Étant donné* (x, Mpk)*, avec* $x \in \mathbb{Z}_p^{\ell}$ *:*

1. *On tire uniformément* $r \in \mathbb{Z}_p$.
2. *On calcule* $C = g^r$, $D = h^r$.
3. *On calcule* $\forall i \in \{1, ..., \ell\}, E_i = \{g^{x_i} \times h_i^r\}_{i=1}^{\ell}$.
4. *On définit* $C_x = (C, D, \{E_i\}_{i=1}^{\ell})$.
5. *On renvoie* $DDH.Encryption\,(x, \text{Mpk}) = C_x$.

Functional Key generation. *Étant donné* (y, Msk)*, avec* $y \in \mathbb{Z}_p^{\ell}$ *:*

1. *On calcule la clé fonctionnelle* $Sk_y = (s_y, t_y) = (s \cdot y, t \cdot y)$.
2. *On renvoie* $DDH.KeyGen\,(y, \text{Msk}) = Sk_y$.

Functional Decryption. *Étant donné* (Mpk, C_x, Sk_y) *:*

1. *On calcule*
$$E_y = \frac{\prod_{i=1}^{\ell} E_i^{y_i}}{C^{s_y} \cdot D^{t_y}}$$
2. *On renvoie* $DDH.Decryption\,(\text{Mpk}, C_x, Sk_y) = \log_g(E_y) = x \cdot y$*, si les conditions d'exactitude sont respectées.*

5.2 Chiffrement fonctionnel de Paillier

Nous nous intéressons dans ce paragraphe à une autre schéma proposé dans [8], qui permet de s'affranchir de la limite du schéma de chiffrement DDH de la partie précédente 5.1. L'idée essentielle du présent schéma est d'exploiter le cryptosystème de Paillier, en construisant un groupe multiplicatif dans lequel le problème du logarithme discret est facile à calculer : ici, on choisit un modulo de type RSA $N = pq$, et on considère le groupe multiplicatif $\mathbb{Z}_{N^2}^*$. Ce groupe contient par construction un sous-groupe d'ordre N généré par $N + 1$ dans lequel le problème du logarithme discret (voir Section 1.7.10) est simple.

Définition du schéma

Ce schéma est un schéma de chiffrement IPFE permettant de calculer le produit scalaire $x \cdot y$ d'un vecteur message $x \in \mathbb{Z}^\ell$ par un vecteur fonction $y \in \mathbb{Z}^\ell$ avec $\|x\|_\infty \leq X$ et $\|y\|_\infty \leq Y$. La variable $\ell \in \mathbb{Z}$ correspond à la taille du message. Les bornes X et Y sont des entiers tels que $XY < N$ et $X, Y < \sqrt{\frac{N}{\ell}}$, avec N le module composite du cryptosystème de Paillier. Le schéma permet de déchiffrer $x \cdot y \mod N$, qui est équivalent à $x \cdot y$ par un choix adapté des bornes X et Y.

Set up. *Étant donné λ le paramètre de sécurité, ℓ la taille du vecteur message x et $X, Y \in \mathbb{N}^*$ respectivement les bornes tel que $\|x\|_\infty \leq X$ et $\|y\|_\infty \leq Y$:*

1. *Soient p, q des nombres premiers de Sophie Germain tel que $p = 2p' + 1, q = 2q' + 1$ avec des nombres premiers $p', q' > 2^{l(\lambda)}$, avec l un polynôme quelconque tel que la factorisation soit 2^λ-difficile.*
2. *Calcul de $N = pq > XY$.*
3. *Tirage uniforme $g' \in \mathbb{Z}^*_{N^2}$.*
4. *Calcul de $g = (g')^{2N}$.*
5. *Tirage de $s \hookleftarrow \mathcal{D}^\ell_\sigma$ suivant une distribution Gaussienne discrète d'écart-type $\sigma > \sqrt{\lambda} \cdot N^{\frac{5}{2}}$.*
6. *Calcul de $h_i = g^{s_i} \mod N^2$.*
7. *Définition la clé secrète maîtresse $\mathrm{Msk} = (s, Y)$ et la clé publique maîtresse $\mathrm{Mpk} = (N, g, \{h_i\}_{i=1}^\ell, X)$.*
8. *Output $PDDH.Setup(\lambda, \ell) = (\mathrm{Msk}, \mathrm{Mpk})$.*

Une fois le Set Up terminé, les nombres p, p', q, q' ne sont plus nécessaires et peuvent être supprimés.

Encryption. *Étant donné (x, Mpk), avec $x \in \mathbb{Z}^\ell$ et $\|x\|_\infty \leq X$:*

1. *Tirage uniforme de $r \in \mathbb{Z}_{\lfloor \frac{N}{4} \rfloor}$.*
2. *Calcul de $C_0 = g^r \mod N^2$ et, $\forall i \in \{1, ..., \ell\}$, $C_i = (1 + y_i N) \cdot h_i^r \mod N^2$.*
3. *Définition de $C_x = (C_0, \{C_i\}_{i=1}^\ell)$.*
4. *Output $PDDH.Encryption\,(x, \mathrm{Mpk}) = C_x$.*

Functional Key generation. *Étant donné (y, Msk), avec $x \in \mathbb{Z}^\ell$ et $\|y\|_\infty \leq Y$:*

1. *Calcul de la clé fonctionnelle* $Sk_y = s \cdot y$.

2. *Output* $PDDH.KeyGen\,(y, \mathrm{Msk}) = Sk_y$.

Functional Decryption. *Étant donné* $(\mathrm{Mpk}, C_x, Sk_y)$ *:*

1. *Calcul du déchiffrement fonctionnel*

$$C_y = \frac{\prod_{i=1}^{\ell} C_i^{y_i}}{C_0^{Sk_y}} \mod N^2.$$

2. *Output* $PDDH.Decryption\,(\mathrm{Mpk}, C_x, Sk_y) = log_{(N+1)}(C_y) = x \cdot y$, *si les conditions d'exactitude sont respectées [8].*

Le produit scalaire calculé est $x \cdot y \mod N$ et est donc défini sur $\mathbb{Z}_N$. L'ajout du modulo est indispensable pour des raisons de sécurité étant donné que les clés secrètes utilisées sont dans des espaces d'entiers comme expliqué dans [8]. Avec des bornes X et Y bien choisies l'application de ce modulo n'impacte pas le résultat obtenu sur $\mathbb{Z}$.

Le calcul suivant donne les conditions d'exactitude du déchiffrement fonctionnel linéaire du schéma basé sur le cryptosystème de Pailler. Par construction, on a $C_y = \frac{\prod_{i=1}^{\ell} C_i^{y_i}}{C_0^{Sk_y}} \mod N^2$, donc en développant on obtient

$$\begin{aligned} C_y &= \frac{\prod_{i=1}^{\ell} ((1 + x_i N)^{y_i} g^{r s_i y_i})}{C_0^{r \sum_{i=1}^{\ell} y_i s_i}} \mod N^2 \\ &= \prod_{i=1}^{\ell} (1 + x_i N)^{y_i} \mod N^2. \end{aligned}$$

On en déduit

$$\begin{aligned} PDDH.Decryption\,(\mathrm{Mpk}, C_x, Sk_y) &= \log_{(N+1)}(C_y) \\ &= \sum_{i=1}^{\ell} y_i \log_{(N+1)}(1 + x_i N) \\ &= x \cdot y. \end{aligned}$$

5.3 Schéma de chiffrement fonctionnel linéaire à mémoire basé sur LWE

On s'intéresse maintenant au deuxième schéma LWE de [8], permettant aussi de calculer le produit scalaire, mais cette fois-ci on rajoute la plus grande valeur mémoire dans la génération de clés fonctionnelles. L'algorithme permet au propriétaire des données de mémoriser dans un état *st* les clés fonctionnelles générées précédemment. L'idée est la suivante : si un utilisateur a accès à trop de clés fonctionnelles indépendantes linéairement, il peut retrouver la base et retrouver les données, à partir des sorties et des clés fonctionnelles. Ce qui, par définition, rend le schéma caduque. La méthode introduite ici permet au propriétaire des données de limiter au maximum le nombre de clés fonctionnelles générées, tout en s'assurant que l'utilisateur ne pourra jamais reconstruire la base complète.

5.3.1 Définition du schéma

Parameters. *Soient* $m, q = p^k$ *pour un entier* k *et* $\alpha \in (0, 1)$.

Set up. *Étant donné* λ *le paramètre de sécurité et* ℓ *la taille du vecteur message* x *et* $n \in \mathbb{Z}$ *:*

1. *Tirage uniforme de* $A \in \mathbb{Z}_q^{m \times n}$.
2. *Tirage de la matrice* $Z \hookleftarrow \tau^{\ell \times m}$ *de taille* $\ell \times m$*, la distribution* τ *est détaillée dans la Section 2.*
3. *Calcul de* $U = ZA \in \mathbb{Z}_q^{\ell \times n}$.
4. *Définition de la clé secrète maîtresse* $\mathrm{Msk} = Z$ *et de la clé publique maîtresse* $\mathrm{Mpk} = (A, U)$.
5. *Output* $NLWE.Setup(\lambda, \ell) = (\mathrm{Msk}, \mathrm{Mpk})$.

Encryption. *Étant donné* (x, Mpk)*, avec* $x \in \mathbb{Z}^\ell$ *et* $\|x\|_\infty < P$ *:*

1. *Tirage uniforme de* $s \in \mathbb{Z}_q^n$.
2. *Tirage du bruit Gaussien* $e_0 \hookleftarrow \mathcal{D}_\sigma^m$ *et* $e_1 \hookleftarrow \mathcal{D}_\sigma^\ell$ *avec* $\sigma = \alpha p$.
3. *Calcul de* $c_0 = sA^T + e_0 \in \mathbb{Z}_q^m$ *et* $c_1 = sU^T + e_1 + p^{k-1}x \in \mathbb{Z}_q^\ell$.
4. *Définition de* $C = (c_0, c_1)$.
5. *Output* $NLWE.Encryption\,(x, \mathrm{Mpk}) = C$.

Functional Key generation. *Étant donné* (y, Msk, st)*, avec* $y \in \mathbb{Z}^\ell$ *et* $\|y\|_\infty < V$ *et st un état interne de l'algorithme de génération de clés fonctionnelles. L'état st peut contenir au plus* ℓ *uplets* $(y_i, \bar{y}_i, Sk_i)$*. Les* x_i *correspondent aux requêtes ayant déjà donné lieu à la génération d'une clé fonctionnelle et le couple* $(\bar{y}_i, Sk_i)$ *correspond aux clés secrètes associées. Si* y *est linéairement indépendant des* y_i *modulo* p*, alors*

1. *Définition de* $\bar{y} = y \in \mathbb{Z}^\ell$*.*
2. *Calcul de* $Sk_y = \bar{y}Z \in \mathbb{Z}^m$*.*
3. *Ajout de* $(y, \bar{y}, Sk_y)$ *à st.*
4. $NLWE.KeyGen\,(y, \mathrm{Msk}) = (\bar{y}, Sk_y)$*.*

Sinon, y *est de la forme* $y = \sum_i k_i y_i \in \mathbb{Z}^m$*, pour* $k_i \in [0, p]$*.*

1. *Définition de* $\bar{y} = \sum_i k_i \bar{x}_i \in \mathbb{Z}^\ell$*.*
2. *Calcul de* $Sk_i = \sum_i k_i z_i \in \mathbb{Z}^m$*.*
3. $NLWE.KeyGen\,(y, \mathrm{Msk}) = (\bar{y}, Sk_y)$*.*

Functional Decryption. *Étant donné* $(\mathrm{Mpk}, C, \bar{y}, Sk_y)$ *:*

1. *Calcul du déchiffrement fonctionnel*

$$C'_y = \bar{y}c_1^T - Sk_y c_0^T \mod p$$

2. *Recherche de la valeur* $C_y \in \mathbb{Z}_p$ *minimisant* $|p^{k-1}C_y - C'_y|$*. En pratique, on peut calculer* $C_y = \lfloor C'_y / p^{k-1} \rceil$*.*
3. *Output* $LWE.Decryption\,(\mathrm{Mpk}, C_x, Sk_y) = C_y = x \cdot y$*, si les conditions d'exactitudes sont respectées.*

5.3.2 Conditions d'exactitude

Lemme 5.3.1. *L'exactitude du déchiffrement fonctionnel LWE à mémoire, tel que décrit ci-dessus, est vérifiée si*

$$|\epsilon| < \frac{1}{2}p^{k-1},$$

avec $\epsilon = ye_1^T - (yZ)e_0^T \in \mathbb{Z}$*.*

Démonstration. Soit $\epsilon = ye_1^T - (yZ)e_0^T \in \mathbb{Z}$. Par construction, on a :

$$\begin{aligned} C'_y &= yc_1^T - Sk_y c_0^T \mod p, \\ &= y\left(s(ZA)^T\right)^T + ye_1^T + p^{k-1}xy^T - (yZ)\cdot(sA^T) - yZe_0^T, \\ &= p^{k-1}x\cdot y + \epsilon. \end{aligned}$$

On a donc bien $LWE.Decryption\,(\mathrm{Mpk}, C_x, Sk_y) = C_y = \lfloor \frac{C'_y}{p^{k-1}} \rceil = x\cdot y$ si $|\epsilon| < \frac{1}{2}p^{k-1}$. □

Deuxième partie

Machine Learning préservant la vie privée

Dans cette seconde partie de la thèse, nous présentons un certain nombre d'expérimentations de Machine Learning couplées à du chiffrement fonctionnel. Nous détaillons tout d'abord la méthodologie, en introduisant des scénarios applicatifs. utiles à ces expérimentations. Dans ces scénarios, Alice désigne le propriétaire de l'ensemble de données privées noté B, dénommé propriétaire des données, et Bob désigne le concepteur de modèle de machine learning, dénommé datascientist de scientifique des données en anglais. Alice et Bob souhaitent concevoir un modèle de machine-learning entraîné sur les données privées, sans que Bob n'est effectivement accès aux informations privées contenues par les données d'Alice.

Chapitre 6

Travaux préparatoires

6.1 Hypothèses et définition des scénarios

Les trois hypothèses suivantes sont supposées tout au long de cette seconde partie de la thèse :

1. Le format de la base de données privées est connue du public et une base de données publiques notée A contenant des données similaires au même format est identifiée et accessible. Cette hypothèse est nécessaire à l'utilisation d'apprentissage par transfert. En particulier, l'étude d'une base de données publiques similaires permettra d'identifier les descripteurs utiles pour la classification.
2. Alice est capable de distinguer les informations privées non partageables des informations publiques partageables parmi ses données. Cette hypothèse est nécessaire car elle permet à Alice de décider quelles fonctions des données il est possible de partager avec Bob.
3. Les informations privées des données d'Alice ne sont pas essentielles pour la tâche à remplir par le modèle de machine-learning à concevoir par Bob. Autrement dit, étant donné un accès hypothétique au sous-ensemble non-privé de l'information contenue par les données d'Alice, Bob est en capacité de concevoir le modèle.

Pour la définition de scénarios applicatifs, nous considérons un processus de conception de modèles de machine-learning basé sur l'apprentissage par transfert en 3 étapes distinctes : le pré-traitement, le cas échéant l'apprentissage sur la base de données publiques et l'apprentissage sur la base de données privées. De fait, l'apprentissage sur la base de données chiffrées par Bob s'effectue sur un sous-ensemble autorisé des données d'Alice : nous nous appuyons sur l'utilisation du chiffrement fonctionnel pour le garantir.

On distingue ici 3 types de fonctions, de pré-traitement ou sous-ensemble de modèle :

1. Les fonctions avec apprentissage, qui dépendent de la base de données utilisée mais sont indépendantes des classes auxquelles sont associées les données.

2. Les fonctions avec apprentissage, qui dépendent de la base de données utilisée et des classes auxquelles sont associées les données.
3. Les fonctions indépendantes de la base de données, qui ne nécessitent pas de calibration ou d'apprentissage.

Dans le premier cas, il est nécessaire d'effectuer un transfert d'apprentissage d'une base de données publiques A accessible à Bob vers la base de données privées B appartenant à Alice. Ce cas est traité par le **Scénario 1**. Dans le deuxième cas, les classes impactent la fonction : dans le cas d'un apprentissage par transfert il est nécessaire que les jeux A et B présentent les mêmes classes. Ce cas est traité par le **Scénario 2**. Notez que lorsque la dépendance de la fonction aux classes reste à déterminer, on préférera expérimenter en premier lieu le **Scénario 1**. Le troisième cas de figure qui ne nécessite pas d'apprentissage par transfert est traité par le **Scénario 3**.

Scénario 1

Alice est la propriétaire d'une base de données sensibles B et il existe une base de données publiques A ou en tout cas accessibles aux concepteurs qui ont des données de la même structure que la base de données B. Cependant, les données de B n'ont pas les mêmes classes que la base de données A. Dans ce scénario, le processus de conception se décompose par les étapes suivantes :

1. En utilisant l'hypothèse 1 de la Section 6.1, Alice et Bob identifient un ensemble de données publiques B dont la structure des données ressemble à celle de l'ensemble de données privées A.
2. Bob conçoit et entraîne une fonction de pré-traitement ou un modèle de machine learning sur B.
3. Bob sélectionne les fonctions qu'il considère comme les meilleures pour encoder A. Il soumet à Alice ces fonctions et les paramètres de résolution de données et de fonctions associées. La partie 6.3 explique, dans le cas où la fonction est un sous-ensemble de modèle de machine learning, le choix de couches à soumettre.
4. En prenant en compte les paramètres de résolution demandés par Bob, Alice chiffre A en $\mathcal{C}_A$ en suivant le ou les schémas appropriés pour une fonction linéaire (5.1) (5.3.1)(3.1) ou quadratique (4.2) et envoie $\mathcal{C}_A$ à Bob.

5. Par l'hypothèse 2 de la Section 6.1, Alice décide des couches qui peuvent être partagées à Bob et envoie les clés fonctionnelles correspondantes $\mathbf{Sk_f}$ à Bob.
6. À l'aide de $\mathcal{C}_A$ et de $\mathbf{Sk_f}$, Bob déchiffre les résultats des fonctions autorisées, en suivant l'étape de déchiffrement du schéma utilisé.
7. A partir des fonctions des données B obtenues, et en utilisant l'hypothèse 3 (cf. Section 6.1), Bob est en mesure de concevoir un modèle d'apprentissage automatique apte à la tâche visée sur les données B.

La Figure 6.1 ci-dessous illustre ce processus dans le cas où la fonction est équivalente aux premières couches d'un modèle de machine learning.

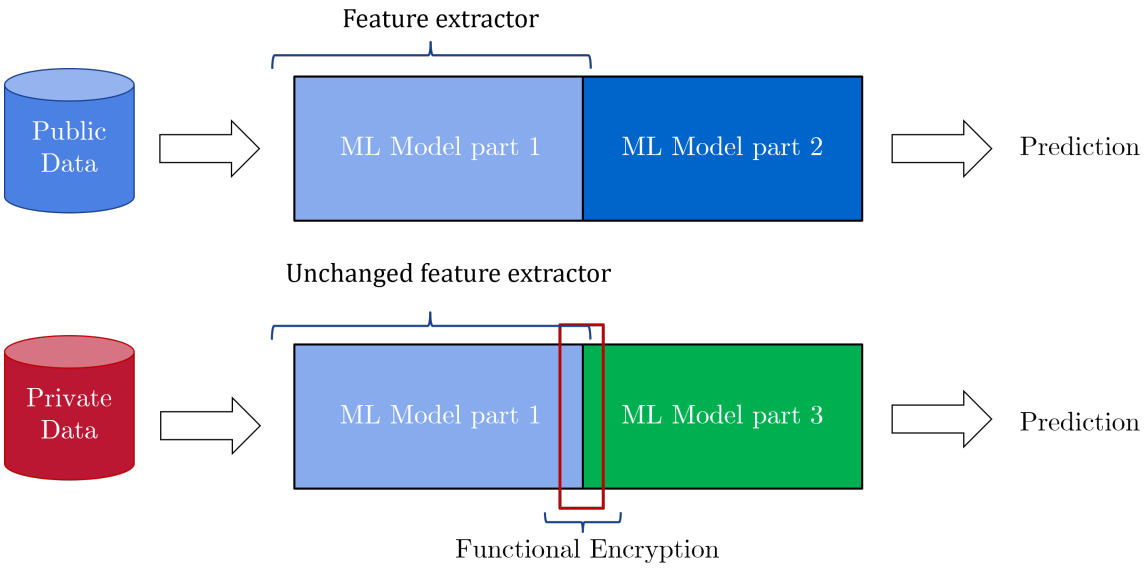

FIGURE 6.1 – Illustration du processus d'apprentissage par transfert d'une fonction apprise par modèle de machine learning

Scénarios 2 et 3

Le **Scénario 2** est identique au Scénario 1. La différence est que les bases de données publiques B et privées A ont les mêmes labels.

Pour le **Scénario 3**, l'apprentissage par transfert n'est pas utile. On démarre donc le processus proposé dans le **Scénario 1** à l'étape 4.

6.2 Apprentissage par transfert

Le transfert d'apprentissage [72] en IA est une technique qui consiste à utiliser les connaissances acquises lors de l'apprentissage d'une tâche pour améliorer les performances d'une autre tâche similaire.

Supposons que nous ayons un modèle d'apprentissage automatique préalablement entraîné sur une tâche A, avec des poids et des paramètres optimisés. Ce modèle est capable de capturer des caractéristiques et des relations entre les données d'entrée et les sorties attendues pour la tâche A. Maintenant, nous souhaitons résoudre une tâche B similaire à la tâche A, mais pour laquelle nous n'avons pas suffisamment de données d'entraînement ou de ressources pour former un modèle à partir de zéro.

Le transfert d'apprentissage consiste à utiliser le modèle préalablement entraîné sur la tâche A comme point de départ pour la tâche B. Cela peut se faire en réutilisant les couches, et donc les poids et paramètres du modèle initial et en les adaptant à la nouvelle tâche.

En utilisant cette approche, le modèle peut bénéficier des connaissances et des représentations apprises lors de la tâche A, ce qui peut accélérer l'apprentissage et améliorer les performances sur la tâche B. Le transfert-learning, souvent désigné par pré-apprentissage, est aujourd'hui communément utilisé et est même parfois considéré comme un pré-requis à l'apprentissage de certains types de modèles comme les Transformers [116].

6.3 Choix de couches du modèle calculé par FE

Notre approche vise à extraire de manière sécurisée une fonction à partir de données chiffrées confidentielles. Cependant, il est essentiel de faire un choix judicieux de cette fonction afin qu'elle puisse être utilisée comme descripteurs classifiants, au sens d'utiles à la tâche de classification, en entrée des modèles à concevoir.

En apprentissage automatique on cherche généralement à soumettre un maximum d'informations en entrée du modèle, en particulier dans le cas d'une conception de modèles d'IA à partir de zéro (from scratch en anglais). En effet, les modèles d'IA sont réputés capables de gérer efficacement de grandes quantités de variables. Cependant, dans notre cas, nous souhaitons pouvoir entraîner une IA en utilisant le moins d'informations possible : le propriétaire des données souhaite limiter au maximum la quantité d'informations partagées concernant ses données. Il est donc préférable que notre protocole n'exige pas

d'informations superflues de sa part. L'objectif est ainsi de définir une fonction calculable par FE permettant d'extraire les informations classifiantes des données tout en réduisant au maximum la quantité d'information globale partagée par le propriétaire des données.

Plusieurs approches sont envisageables pour répondre à cette problématique. La première consiste à appliquer des techniques de réduction de dimension connues du Machine Learning pour l'extraction de caractéristiques classifiantes des données. Cela permet de s'assurer que les fonctions extraites par cette méthode sont pertinentes pour la classification tout en réduisant l'information partagée par le propriétaire des données. Dans cette optique, nous proposons des expérimentations d'application du FE à l'ACP (7.1) (7.1.2) et à la LDA (7.2) (7.2.2).

Une autre approche consiste à utiliser une réduction de dimension calculée par le modèle lui même. En effet, de manière générale, la structure d'un modèle de machine learning réduit naturellement la dimension des données à mesure qu'elles sont transformées. Cette notion s'applique à tous les modèles, bien que les modèles de réseau de neurones en soient le parfait exemple. Au début du modèle les données sont peu transformées : il y a plus de complexité et donc plus d'informations. Au fur et à mesure que l'on progresse dans le modèle, l'information est filtrée pour ne préserver que sa part d'information utile à la classification. En ce sens, nous proposons des expérimentations d'application du FE à un CNN linéaire(7.3) (7.3.2) et un CNN quadratique (8.2)(8.2.2) .

Chapitre 7

Expérimentations de calibration basées sur le chiffrement fonctionnel linéaire pour le pré-traitement d'un jeu de données

---

L'objectif de cette partie est de vérifier la faisabilité d'applications concrètes de nos schémas de chiffrement fonctionnel dans le processus de conception de modèles d'apprentissage automatique (machine learning en anglais). Nous présentons les résultats de tests de plusieurs méthodes de machine learning : l'analyse en composantes principales, l'analyse discriminante linéaire et les réseaux de neurones.

Nos expérimentations s'appuient sur deux bases de données : la base de données MNIST et la base de données ASM présentées ci-dessous.

Présentation de la base de données MNIST

La base de données MNIST [75] est une base de données publiques largement utilisée dans le domaine du machine learning, notamment pour l'évaluation de modèle de reconnaissance de formes. Cette base de données est composée d'images de chiffres manuscrits allant de 0 à 9. Elle comprend 60 000 images pour l'apprentissage et 10 000 images pour les tests, toutes ayant une taille de 28x28 pixels.

Nous avons sélectionné cette base de données, qui est facilement accessible, car elle est couramment utilisée comme référence dans le domaine du machine learning. Par ailleurs, elle est pertinente pour représenter nos cas d'utilisations industrielles axés sur le traitement d'images et est compatible avec les différents scénarios que nous définissons en Section II. En particulier, dans le cadre d'apprentissage par transfert (transfer learning en anglais), un découpage arbitraire de la base de données MNIST nous a permis de si-

muler des cas où le concepteur a accès à un jeu de données publiques ayant une structure similaire mais des étiquettes (labels en anglais) différentes de celles d'un jeu de données cible.

Présentation du jeu de données Acoustique Sous-Marine (ASM)

Le jeu de données ASM est constitué de données réelles issues d'un sonar passif militaire américain du 20-ème siècle, mises à disposition par l'Université de San Francisco [**San Francisco**]. Étiqueté manuellement, ce jeu de données contient en particulier des enregistrements d'émissions sonar (transitoires acoustiques générés par un sonar actif) notées "ES". Nous avons décomposé de façon aléatoire ce jeu de données en un jeu d'apprentissage de 602 exemples, dont 78 exemples "ES" dits positifs, et un jeu de test de 151 exemples, dont 19 positifs. Les expérimentations rapportées utilisant ce jeu de données s'intéressent à la tâche de classification binaire des cas positifs "ES" et négatifs.

7.1 Analyse en Composantes Principales couplée au chiffrement fonctionnel pour la conception d'un réseau de neurones

7.1.1 Contexte et objectifs de l'expérimentation

Couramment utilisée dans le cadre de travaux de data science, l'Analyse en Composantes Principales (ACP) est une technique de réduction de dimension linéaire permettant d'exprimer un maximum d'information en un minimum de dimension. Nous nous intéressons dans cette section à son utilisation dans le cadre du chiffrement fonctionnel.

Comme expliqué ci-après dans le paragraphe (7.1.2), l'ACP permet d'obtenir une matrice de passage qui, appliquée aux données, réalise cette réduction de dimension. En exploitant les schémas détaillés aux paragraphes (2.1), (3.1)et (4.2), nous avons donc cherché à transformer de façon chiffrée des données issues de la base MNIST par produit avec cette matrice de passage.

Sauf mention contraire, tous les résultats numériques de mesures rapportés ont été obtenus par moyenne de 20 échantillons indépendants.

7.1.2 Description de l'Analyse en Composantes Principales

L'Analyse en Composantes Principales (ACP) est une méthode statistique qui permet de réduire la dimensionnalité d'un ensemble de données en projetant ses variables dans un espace de dimension inférieure. L'objectif de l'ACP est de trouver un petit nombre de variables non corrélées, nommées composantes principales, expliquant un maximum de la variance totale de l'ensemble de données. Elle est utilisée pour résumer et explorer de grands ensembles de données, et pour identifier les relations entre les variables.

L'application de l'ACP à un jeu de données se fait à l'aide d'un changement de base par produit matriciel. La matrice de passage est calculée à partir de la matrice des données, traitée de la manière suivante :

1. Centrage des données : on soustrait à chaque variable sa moyenne afin de centrer les données autour de 0.
2. Réduction des données : on divise chaque variable par son écart-type afin de mettre les données à la même échelle.
3. Calcul de la matrice de covariance des données.
4. Calcul des vecteurs propres et des valeurs propres de la matrice de covariance.
5. Construction de la matrice de passage par concaténation des vecteurs propres en colonnes, ordonnés selon leurs valeurs propres classées dans l'ordre décroissant.

La matrice de passage ainsi obtenue est une matrice carrée permettant de projeter les données initiales sur les axes principaux de l'ACP par produit matriciel. Cela permet dans le cas d'un pré-traitement pour le machine learning de simplifier la représentation des données : empiriquement, on constate souvent que les premières composantes principales sont les plus utiles aux tâches de classification. Représenter les données initiales par leurs projections sur les premières composantes principales a tendance à faciliter et accélérer l'apprentissage d'un modèle.

7.1.3 Expérimentations et recherche de paramètres IA

Nous avons utilisé le chiffrement fonctionnel afin de calculer les produits scalaires nécessaires pour effectuer la réduction de dimension. En effet, chaque produit matriciel réalisé lors de la réduction peut être réinterprété comme plusieurs produits scalaires.

Dans cette expérimentation, nous proposons une approche permettant de choisir la sécurité du schéma, tout en maîtrisant les pertes de performance induites par le protocole

et l'utilisation du chiffrement fonctionnel. Nous présentons des jeux de paramètres pertinents pour chaque cas d'utilisation, c'est-à-dire des paramètres permettant d'obtenir un compromis performance/sécurité optimisé.

La méthode utilisée pour l'expérimentation implique de réduire les données MNIST en utilisant l'ACP.

Formellement, pour une image $x \in \mathbb{Z}^{\ell}$ et une matrice de passage $M \in \mathbb{Z}^{\ell \times Q}$ restreinte à Q composantes principales, on calcule le produit matriciel $x \times M$ en effectuant Q produits scalaires $x \cdot M_i$, avec $i \in \{0, ..., Q-1\}$. Chaque vecteur M_i correspond à une colonne de la matrice ACP et chaque résultat de produit scalaire ainsi obtenu correspond à une composante principale de l'image x. Chaque composante principale est alors considérée comme un descripteur des données et peut être utilisée comme entrée d'un réseau de neurones.

Cette expérimentation est basée sur le scénario 1 décrit dans la partie (II). D'abord, nous avons séparé l'ensemble de données MNIST en deux parties : un premier jeu de données A contenant les images étiquetées de 0 à 4 est considéré comme la base de données publiques, et un second jeu de données B contenant les images étiquetées de 5 à 9 est considéré comme la base de données privées cible. En suivant le scénario 1 décrit en Section (II), nous avons réalisé un apprentissage par transfert sécurisé par FE du jeu A vers le jeu B.

L'objectif de cette expérimentation était de mesurer la perte résultante de ce processus et d'explorer une méthode de sélection de paramètres de FE permettant d'obtenir un compromis satisfaisant entre les performances du modèle de machine learning finalement obtenu et le niveau de sécurité du schéma de chiffrement utilisé.

En ce sens, nous avons évalué différentes pertes de performance dues au protocole en fonction des paramètres choisis, par rapport à un apprentissage standard, soit sans restriction d'accès sur le jeu de données B. Nous avons identifié à ce stade que les pertes de performances peuvent être de 3 natures différentes, étudiées séparément.

En premier lieu, nous avons étudié les pertes dues à l'utilisation de l'apprentissage par transfert en comparant les performances d'un modèle entraîné sans contrainte sur le jeu de données B avec celles d'un modèle entraîné par transfert. Ensuite nous avons effectué des tests afin de déterminer des paramètres pertinents en cherchant les minimums de résolutions de données et de fonctions afin d'observer un minimum de pertes de performances du modèle. Enfin, nous avons méthodiquement étudié les pertes dues à la restriction du nombre de composantes ACP calculées, dans un objectif de minimiser le

nombre de fonctions partagées par le propriétaire des données.

Évaluation des performances de référence

L'architecture du modèle utilisé est décrite dans la Figure 7.2.5. Tout au long de cette expérimentation, cette architecture et les paramètres d'apprentissage utilisés n'ont pas été modifiés afin d'éviter l'introduction de biais. Également, lors des applications de l'ACP de cette expérimentation, nous avons arbitrairement choisi de ne garder que les dimensions expliquant au minimum 1% de la variance totale.

À titre de comparaison, nous avons tout d'abord mesuré les performances d'un modèle de machine learning entraîné sans contraintes, composé d'un réseau de neurones précédé d'une ACP, sur la base de données B. Plus précisément, nous avons construit une matrice de passage ACP M_B, comme expliqué dans la partie 7.1.2. Seules les 18 premières composantes principales de cette matrice de passage M_B expliquent au moins 1% de la variance totale, pour un total de 68% de variance totale expliquée. Nous avons calculé, pour chaque donnée du jeu B, 18 descripteurs classés par ordre croissant de variance expliquée. Ces descripteurs ont ensuite été utilisés comme entrées pour l'apprentissage d'un réseau de neurones tel que défini en partie 7.2.5. Les performances mesurées sont rapportées via la métrique d'exactitude (accuracy en anglais), sur la Table 7.1. La performance $P_R = 98.1\%$ obtenue nous a servie de référence, et peut être considérée comme le maximum atteignable de performance de ce type de modèle sur la base de données B.

Pertes dues à l'utilisation de l'apprentissage par transfert

L'étape suivante a consisté à évaluer les performances d'un modèle ayant la même architecture, mais conçu à l'aide de l'apprentissage par transfert. Pour ce faire, nous avons utilisé la matrice de passage ACP M_A calculée à partir de la base de données A. Comme pour M_B, seules les 18 premières composantes principales de cette matrice de passage M_A expliquent au moins 1% de la variance totale, pour un total de 68% de variance totale expliquée. Nous avons appliqué cette matrice M_A à la base de données B pour effectuer le changement de base. Nous avons ensuite entraîné un réseau de neurones (cf. 7.2.5) en utilisant les 18 premiers descripteurs obtenus. Nous avons mesuré la performance $P_T = 98.0\%$ et nous avons évalué la perte de performance due au transfert en calculant la différence $P_R - P_T = 0.1\%$. Cette perte s'explique par la désadaptation induite par le fait que la matrice de passage M_A utilisée n'a pas été calculée sur le jeu de données cible

B. Cependant, étant donné que le jeu de données publiques d'entraînement présente une structure similaire, elle est minime.

Pertes dues aux arrondis du chiffrement fonctionnel

Le deuxième type de pertes est du aux arrondis inhérents au FE. Par construction, les schémas de chiffrement fonctionnel utilisés (5.2),(2.1) et (3.1) s'appliquent dans des espaces mathématiques d'entiers et, par conséquent, les données d'entrées et la fonction doivent être respectivement arrondies selon une résolution fixée par avance.

Nous avons procédé à l'entraînement de modèles avec la même architecture, en effectuant cette fois-ci le changement de base calculé par ACP selon du chiffrement fonctionnel linéaire. Trois schémas différents ont été utilisés (5.2), (2.1) et (3.1) et le choix des paramètres correspondants à chaque schéma, et donc à chaque modèle, est expliqué en deuxième partie de l'expérimentation, dans la Section 7.1.4.

Ici, les arrondis se sont donc portés sur les données B en entrée et sur la matrice M_A. Il faut noter que, plus la résolution choisie est grande, avec donc une erreur d'arrondi diminuée, plus les paramètres du FE sont contraints, impactant négativement la sécurité ou le temps de calcul. Inversement, des résolutions trop faibles, avec donc des erreurs d'arrondis trop importantes, nuiront aux performances du modèle entraîné ensuite. Il faut donc chercher un compromis acceptable.

Afin d'étudier l'impact des arrondis sur les performances du modèle, nous avons considéré des résolutions allant de 1 à 6 bits. Dans un premier temps, la performance du modèle a été mesurée en appliquant un arrondi uniquement aux données et non à la fonction. Cette approche a permis de mesurer l'impact de l'arrondi des données sur les performances du modèle. Nous avons donc appliqué un arrondi aux données B, suivi d'un changement de base à l'aide de la matrice M_A non arrondie, et enfin nous avons entraîné le modèle de réseau de neurones à partir des descripteurs obtenus. Les performances mesurées sont rapportées dans la Table 7.1 et sur la Figure 7.1.

On peut remarquer qu'à partir de 2 bits de résolution, l'impact d'un plus petit arrondi sur les performances du modèle est minime, alors qu'en dessous d'1 bit de résolution, les performances sont dégradées. Nous avons donc décidé de fixer la résolution de l'image à 2 bits pour ce cas d'usage.

Ensuite, nous avons répété l'expérience en effectuant un arrondi sur la fonction. Nous avons appliqué un arrondi à la matrice M_A, puis nous avons effectué le changement de base sur les données B non arrondies. Enfin, nous avons entraîné le modèle de réseau de

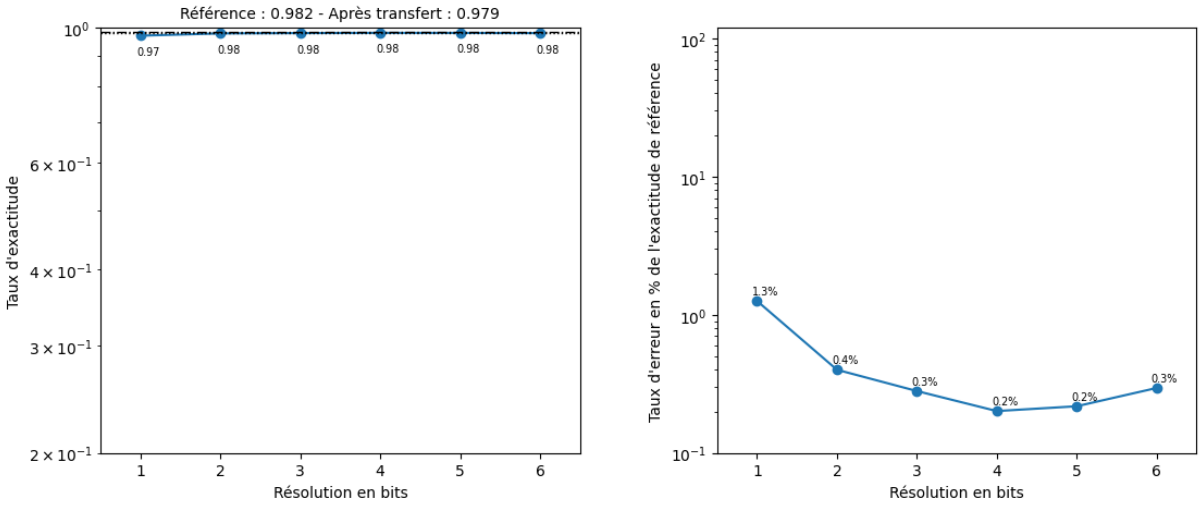

FIGURE 7.1 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution des données après ACP (à gauche) et perte de performance par rapport à la référence (à droite).

neurones en utilisant les descripteurs obtenus. Les performances mesurées sont rapportées dans la Table 7.1 et sur la Figure 7.2.

Il apparaît qu'avec 3 bits de résolution ou moins, le modèle présente des performances dégradées, et qu'à partir de 4 bits de précision, l'impact de l'arrondi sur les performances du modèle est négligeable. Nous avons donc décidé de fixer la précision de l'arrondi sur la fonction à 4 bits de résolution pour ce cas d'usage.

A des fins de vérification, nous avons effectué une dernière expérience sur l'arrondi pour confirmer les résultats obtenus : les valeurs de résolutions ont été choisies indépendamment, afin de confirmer que les pertes de performance du modèle sont négligeables une fois ces deux paramètres combinés. Pour cela, nous avons fixé l'arrondi pour les données à 2 bits de résolution, tout en faisant varier l'arrondi pour la fonction. Autrement dit, nous avons appliqué un arrondi à la matrice M_A puis avons effectué le changement de base sur les données B arrondies à 2 bits de précision. Les performances mesurées sont rapportées dans la Table 7.1 et sur la Figure 7.3.

Nous avons mesuré la performance $P_A = 97.3\%$ et avons évalué la perte de performance due à l'arrondi en calculant la différence $P_T - P_A = 0.7\%$. On constate qu'avec une résolution de 2 bits de précision pour les images en entrée B et de 4 bits de précision pour la matrice de passage M_A, les performances sont peu impactées. Cela a permis de confirmer le choix de ces paramètres pour le restant de l'expérimentation.

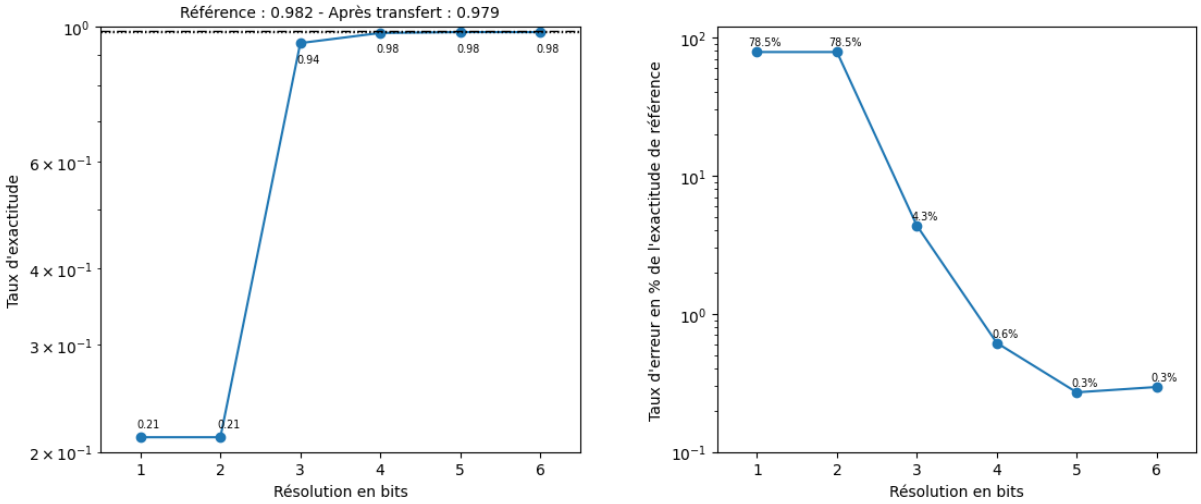

FIGURE 7.2 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution de la fonction après ACP (à gauche) et perte de performance par rapport à la référence (à droite).

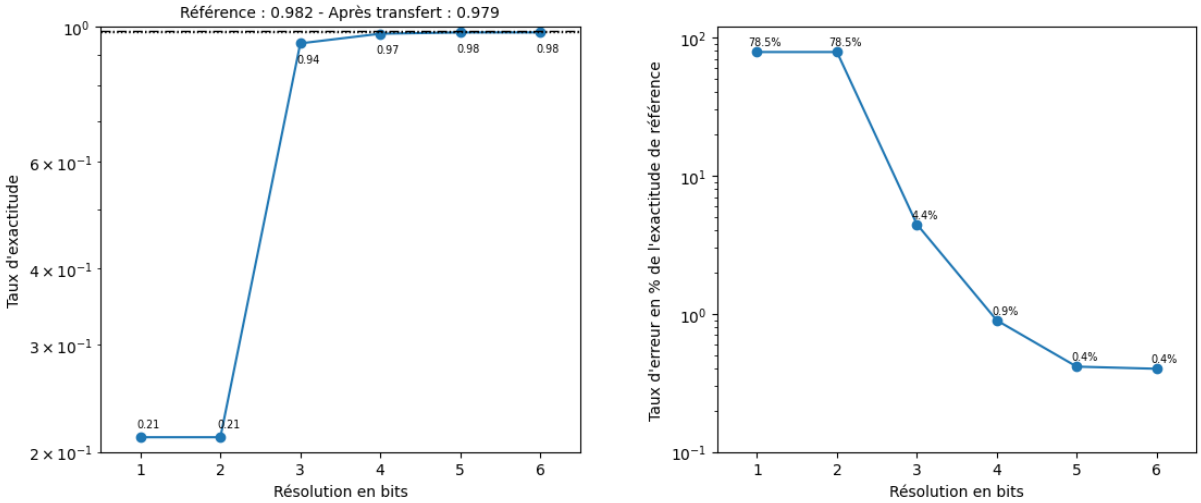

FIGURE 7.3 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution de la fonction après ACP avec une résolution fixée à 2 bits pour les données (à gauche) et perte de performance par rapport à la référence (à droite).

Pertes dues à la restriction du nombre de composantes ACP calculées

Le dernier type de pertes identifié est lié à la restriction du nombre de composantes ACP calculées. En effet notre protocole implique que le propriétaire des données ne peut potentiellement donner accès qu'à une part des descripteurs au concepteur de l'IA, comme expliqué dans la partie II. Le cas échéant, il faut être en mesure d'évaluer la perte de performance qu'implique le non-accès à certains descripteurs.

Pour cela, nous avons appliqué les mêmes transformations qu'au cours de l'expérimentation précédente sur la base de données B cible : nous avons effectué un apprentissage par transfert et un arrondi à une résolution de 2 bits pour les images et à une résolution de 4 bits pour la matrice de passage. Nous avons ensuite entraîné indépendamment 20 modèles de réseau de neurones à partir de respectivement 1 à 20 descripteurs. Les modèles sont tous basés sur l'architecture décrite sur la Figure 7.2.5, seule la taille de l'entrée varie selon le nombre de descripteurs sélectionnés. Les descripteurs accessibles sont ici ajoutés par ordre croissant de part de variance totale expliquée. Les descripteurs les plus porteurs d'informations classifiantes, au sens d'utile à la tâche de classification, sont ainsi a priori ajoutés en premier. Les performances mesurées sont rapportées dans la Table 7.1 et sur la Figure 7.4.

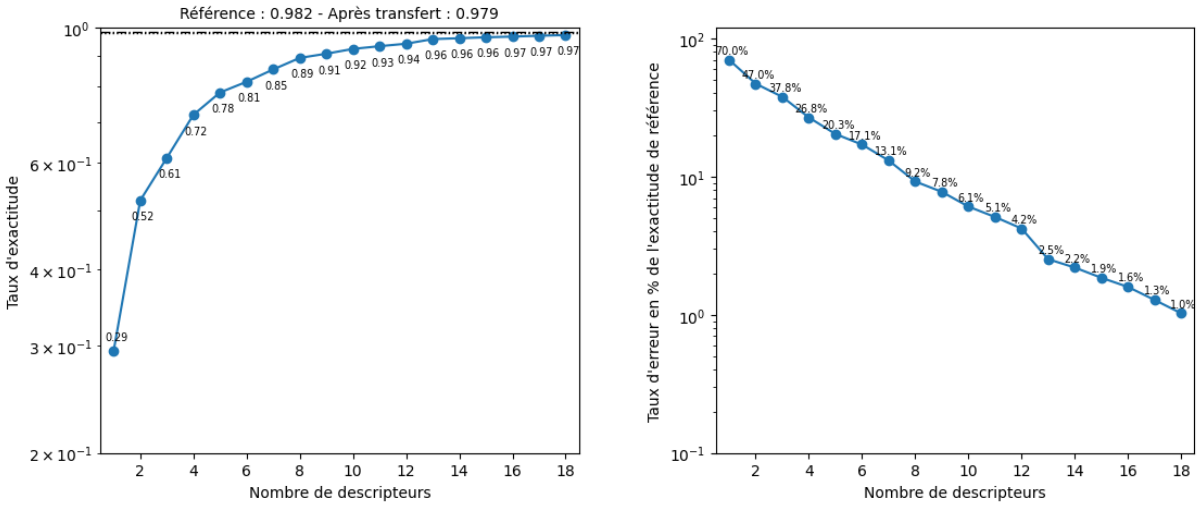

FIGURE 7.4 – Taux d'exactitude mesuré en fonction du nombre de composantes principales.

Cet ordonnancement arbitraire nous a semblé pertinent : d'un point de vue scénaris-

tique, cela permet au propriétaire des données de maximiser la performance du modèle en diffusant un minimum de fonctions des données. Dans un scénario d'application réelle, il n'y a pas nécessairement de corrélation entre la sensibilité de la fonction calculée et la variance expliquée qu'elle contient : le propriétaire des données communique au concepteur de l'IA les fonctions auxquelles il souhaite lui donner accès, sans garantie de variance totale expliquée. En retour, le concepteur de l'IA peut estimer les éventuelles pertes de performance dues à la restriction d'information. Le détail de ce protocole est présenté dans la Section II.

7.1.4 Recherche paramétrique pour le chiffrement fonctionnel

Ayant identifié les paramètres acceptables pour les performances de l'IA, nous déterminons les paramètres correspondants pour le schéma de chiffrement fonctionnel linéaire 3.1. L'arrondi pour la résolution en bits de l'image correspond à la variable $B_x = 2^2$ du schéma et l'arrondi en bits de précision pour la fonction correspond à la variable $B_y = 2^4$. De plus la taille du vecteur x en entrée correspond à la taille de l'image d'entrée 28×28 aplatie. On choisit donc pour le schéma (3.1) les paramètres suivants : $B_x = 2, B_y = 4$ et $w = 784$. La probabilité d'exactitude minimum recherchée est $\mathbb{P}^{(1)} \geq 0.995$. Nous proposons deux jeux de paramètres correspondants à deux niveaux de sécurité $S_1 = 80$ bits et $S_2 = 120$ bits. Nous utilisons pour cela la recherche paramétrique décrite dans la partie 3.5. Nous obtenons les paramètres suivants pour le niveau de sécurité S_1 :

$$\begin{cases} \text{Sécurité} = 80 \text{ bits} \\ n = 1024 \\ \log(p) = 168 \\ \sigma = 158456325028528675187087900672 \\ \sigma_1 = \alpha_1 p_2 = 225 \\ \sigma_2 = \alpha_2 p_2 = 10222334440240441893669414648152064001 \\ \sigma_3 = \alpha_3 p_2 = 204446688804808837873388292296304128000 \end{cases}$$

Et les paramètres suivants pour le niveau de sécurité S_2 :

$$\begin{cases} \text{Sécurité} = 120 \text{ bits} \\ n = 2048 \\ \log(p) = 117 \\ \sigma = 17592186044416 \\ \sigma_1 = \alpha_1 p_2 = 225 \\ \sigma_2 = \alpha_2 p_2 = 2269814212194729984001 \\ \sigma_3 = \alpha_3 p_2 = 4539628424389459968000 \end{cases}.$$

7.1.5 Architecture du modèle

Voici l'architecture utilisée pour les expérimentations (7.1)(7.2). La variable C correspond au nombre classes prédites par le modèle selon l'expérimentation.

Architecture Keras			
Layer	Taille de la sortie	Paramètres	Fonction d'activation
Dense	(None, 512)	2560	ReLu
Dropout	(None, 512)	0	None
Dense	(None, 512)	26265	Relu
Dropout	(None, 512)	0	None
Dense	(None, C)	5130	Softmax

Tous les paramètres d'apprentissage sont laissés par défaut sauf :

1. Taille de batch = 128
2. Loss = categorical crossentropy
3. Optimiseur = RMSprop [110]

7.1.6 Performances des modèles entraînés suite à une ACP

Les performances mesurées sont rapportées par la Table 7.1.

TABLE 7.1 – Performance des modèles ACP

Transfert	Résolution des données	Résolution de la fonction	Descripteurs	Taux d'exactitude
Non	Pas d'arrondi	Pas d'arrondi	18	98.1% (±0.1)
Oui	Pas d'arrondi	Pas d'arrondi	18	98.0% (±2)
Oui	0 bits	Pas d'arrondi	18	21.1% (±0)
Oui	1 bits	Pas d'arrondi	18	97.0% (±0.2)
Oui	2 bits	Pas d'arrondi	18	98.8% (±0.1)
Oui	3 bits	Pas d'arrondi	18	98.0% (±0.2)
Oui	4 bits	Pas d'arrondi	18	98.0% (±0.1)
Oui	5 bits	Pas d'arrondi	18	98.0% (±0.1)
Oui	6 bits	Pas d'arrondi	18	98.0% (±0.2)
Oui	Pas d'arrondi	0 bits	18	21.1% (±0)
Oui	Pas d'arrondi	1 bits	18	21.1% (±0)
Oui	Pas d'arrondi	2 bits	18	21.1% (±0)
Oui	Pas d'arrondi	3 bits	18	93.9% (±0.1
Oui	Pas d'arrondi	4 bits	18	97.6% (±0.2)
Oui	Pas d'arrondi	5 bits	18	97.9% (±0.1)
Oui	Pas d'arrondi	6 bits	18	97.9% (±0.2)
Oui	2 bits	0 bits	18	21.1% (±0)
Oui	2 bits	1 bits	18	21.1% (±0)
Oui	2 bits	2 bits	18	21.1% (±0)
Oui	2 bits	3 bits	18	93.8% (±0.1)
Oui	2 bits	4 bits	18	97.3% (±0.2)
Oui	2 bits	5 bits	18	97.8% (±0.1)
Oui	2 bits	6 bits	18	97.8% (±0.2)
Oui	2 bits	4 bits	1	29.4% (±0.5)
Oui	2 bits	4 bits	2	52.0% (±0.3)
Oui	2 bits	4 bits	3	61.1% (±0.4)
Oui	2 bits	4 bits	4	71.9% (±0.5)
Oui	2 bits	4 bits	5	78.2% (±0.2)
Oui	2 bits	4 bits	6	81.4% (±0.4)
Oui	2 bits	4 bits	7	85.3% (±0.4)
Oui	2 bits	4 bits	8	89.1% (±0.2)
Oui	2 bits	4 bits	9	90.6% (±0.2)
Oui	2 bits	4 bits	10	92.2% (±0.2)
Oui	2 bits	4 bits	11	93.2% (±0.2)
Oui	2 bits	4 bits	12	94.0% (±0.9)
Oui	2 bits	4 bits	13	95.7% (±0.2)
Oui	2 bits	4 bits	14	96.0% (±0.2)
Oui	2 bits	4 bits	15	96.4% (±0.2)
Oui	2 bits	4 bits	16	96.6% (±0.1)
Oui	2 bits	4 bits	17	96.9% (±0.1)
Oui	2 bits	4 bits	18	97.2% (±0.1)

7.2 Analyse discriminante linéaire couplée au chiffrement fonctionnel pour la conception d'un réseau de neurones

7.2.1 Contexte et objectifs de l'expérimentation

Similaire fonctionnellement à l'ACP, l'analyse discriminante linéaire (LDA) est également une technique de machine learning utilisée pour la classification de données. Elle vise à trouver une combinaison linéaire des variables décrivant un ensemble de données qui maximise la séparation entre différentes classes visées. L'objectif de l'analyse discriminante linéaire est donc, comme l'ACP, de réduire la dimensionnalité des données, tout en conservant autant d'informations discriminantes que possible. Nous nous intéressons ici à son couplage au chiffrement fonctionnel.

En exploitant les schémas détaillés précédemment aux paragraphes (5.2), (3.1) et (2.1), nous avons donc cherché à transformer des données issues de la base MNIST par produit avec la matrice de passage calculée par analyse discriminante linéaire, telle que décrite ci-dessous au paragraphe (7.2.2).

Sauf mention contraire, tous les résultats numériques de mesures rapportés ont été obtenus par moyenne de 20 échantillons indépendants.

7.2.2 Description de l'analyse discriminante linéaire

La réduction de dimension LDA (Linear Discriminant Analysis) fonctionne de la manière suivante [109] :

1. Soit un ensemble de N échantillons, représentés par $X = [x_i]_{i=1}^{N}$ la matrice de dimension $N \times M$, où chaque échantillon x_i est un vecteur de dimension M et est associé à une classe $y_i \in Y$ parmi C classes possibles. On dénote X_i le sous-ensemble de X dont les échantillons sont associés à la classe y_i et n_i le nombre d'échantillons de chaque X_i respectivement.
2. La formule suivante est utilisée pour calculer la moyenne μ_i des X_i de chaque classe :

$$\mu_i = \frac{1}{n_i} \sum_{x \in X_i} x$$

.

3. La formule suivante est utilisée pour calculer la moyenne μ totale de X :

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

.

4. La matrice inter-classes S_B est calculée de la manière suivante :

$$S_B = \sum_{i=1}^{C} n_i(\mu_i - \mu)(\mu_i - \mu)^T.$$

5. La matrice intra-classes S_W est calculée de la manière suivante :

$$S_W = \sum_{i=1}^{C}\sum_{j=1}^{n_i}(X_{ij} - \mu_i)(X_{ij} - \mu_i)^T$$

où X_{ij} désigne le j-ème échantillon de X_i.

6. En utilisant S_B et S_W, la matrice W qui maximise la formule de Fisher peut-être calculée par $W = S_W^{-1}S_B$. On calcule alors les valeurs propres λ et les vecteurs propres V de W.
7. Les vecteurs propres sont triés de manière décroissante selon leurs valeurs propres respectives. Ensuite, les k vecteurs propres sont utilisés comme espace de dimension réduite noté V_k.
8. Il est alors possible de projeter les échantillons originaux X dans un espace de dimension réduite par la multiplication matricielle XV_k .

7.2.3 Expérimentations et recherche de paramètres IA

De la même manière que dans l'expérimentation de l'ACP rapportée en Section 7.1, nous avons cherché à utiliser le chiffrement fonctionnel pour effectuer une réduction de dimension par application d'une matrice de passage LDA en décomposant le produit matriciel en de multiples déchiffrements de résultats de produits scalaires. Nous nous plaçons de fait dans les mêmes conditions expérimentales et, sauf mention contraire, les architectures, paramètres et protocoles de mesure sont identiques.

Les directions discriminantes, équivalent LDA des composantes principales obtenues par ACP, sont calculées en effectuant un produit scalaire avec chaque image. Formellement, pour une image $x \in \mathbb{Z}^\ell$ et une matrice de passage LDA $M \in \mathbb{Z}^{\ell \times Q}$ restreinte à Q

directions discriminantes, on calcule le produit matriciel $x \times M$ en effectuant Q produits scalaires $x \cdot M_i$, avec $i \in \{0, ..., Q-1\}$. Les descripteurs résultant de ces opérations peuvent ensuite être utilisés comme entrées d'un réseau de neurones. La transformation par LDA réduit ici la dimension sans nécessiter de choix arbitraire de nombre de directions discriminantes : nous conservons les 4 composantes LDA calculées. Nous avons tout d'abord suivi le premier scénario 1 décrit dans la Section II en adoptant la même séparation du jeu de données MNIST en deux jeux A et B (7.1). La performance de référence $P_R = 92.6\%$ d'un modèle entraîné sans contraintes est rapportée via la métrique d'exactitude dans la Table 7.1.

En utilisant la matrice de passage LDA M_A calculée à partir de la base de données A pour transformation du jeu de données B, nous avons mesuré la performance $P_T = 61.2\%$, et donc une perte de performance due au transfert significative $P_R - P_T = 31.4\%$. Nous expliquons cette perte par la désadaptation induite par le fait que la matrice de passage M_A utilisée n'a pas été calculée sur le jeu de données cible B : la LDA n'est par conception pas adaptée à l'apprentissage par transfert lorsque les classes des deux jeux de données sont différentes. Face à ce constat, nous avons donc décidé de suivre le scénario 2, décrit en Section II, pour la suite de cette expérimentation : le scénario 2 propose une configuration où le jeu de données publiques, accessible au concepteur, et le jeu de données privées du propriétaire ont les mêmes classes. En dénotant dorénavant A et B deux moitiés distinctes du jeu de données MNIST choisies aléatoirement, nous recommençons l'expérimentation avec le même objectif de mesurer les trois types de pertes de performance déjà identifiées : celles dues au transfert, celles dues aux arrondis et celles dues à la restriction du nombre de directions discriminantes déchiffrables.

Évaluation des performances de référence

Notre référence pour le scénario 2 est construite de la même manière que pour le scénario 1 : après transformation du jeu de données B par application de la matrice LDA M_B calculée sur le jeu de données B, nous avons entraîné un réseau de neurones (cf. 7.2.5) et mesuré ses performances sur la base de données B. La performance mesurée P_R est rapportée dans la Table 7.1 ; elle est considérée comme le maximum atteignable de performance de ce type de modèle sur la base de données B.

Pertes dues aux arrondis du chiffrement fonctionnel

Pour mesurer la perte due aux arrondis inhérents au FE, nous avons procédé à l'entraînement de modèles d'architectures identiques (cf. 7.2.5), en effectuant cette fois-ci le changement de base calculé par LDA par chiffrement fonctionnel linéaire. Les trois schémas (5.2), (2.1), (3) ont été expérimentés et le choix de leurs paramètres correspondants est expliqué en deuxième partie de l'expérimentation.

D'abord, nous étudions l'impact de l'arrondi des données B en entrée et mesurons les performances du modèle final sur les directions discriminantes déchiffrées pour des résolutions allant de 1 à 8 bits. Les performances mesurées sont rapportées dans la Table 7.2 et sur la Figure 7.5.

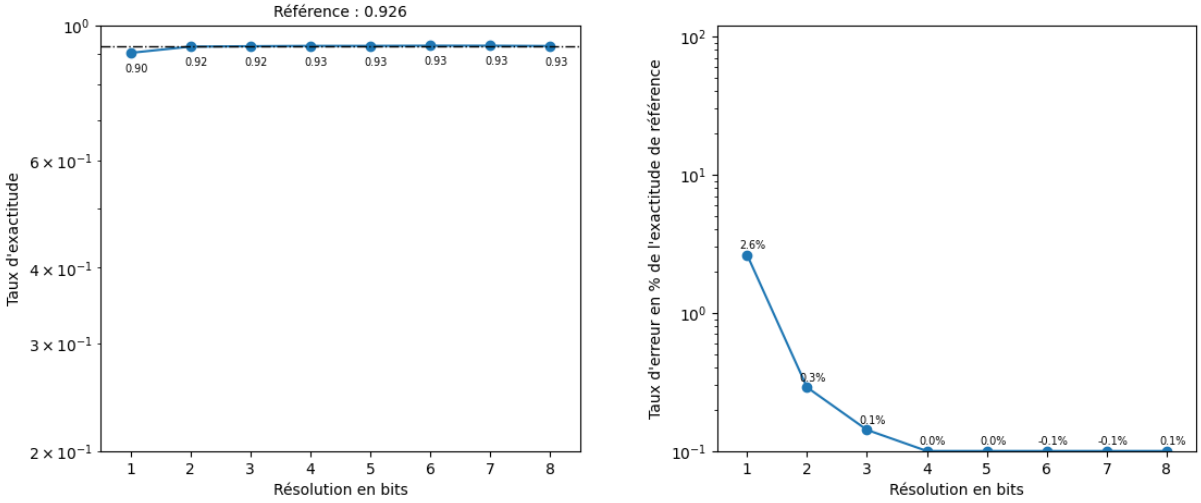

FIGURE 7.5 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution des données après LDA (à gauche) et perte de performance par rapport à la référence (à droite).

On remarque qu'à partir de 2 bits l'impact sur les performances du modèle est minime, ce qui permet raisonnablement de fixer la résolution de l'image à 2 bits pour ce cas d'usage.

Par ailleurs, sans fixer pour le moment la résolution des données B à 2 bits, nous avons répété l'expérience pour étudier l'impact d'un arrondi sur la matrice M_A. Les performances mesurées sont rapportées dans la Table 7.2 et sur la Figure 7.6.

On peut constater qu'avec 4 bits de résolution ou moins, le modèle présente des performances dégradées, et qu'à partir de 5 bits de précision, l'impact de l'arrondi sur les performances du modèle est négligeable. Nous avons donc choisi de fixer la précision de la fonction à 4 bits de résolution pour ce cas d'usage.

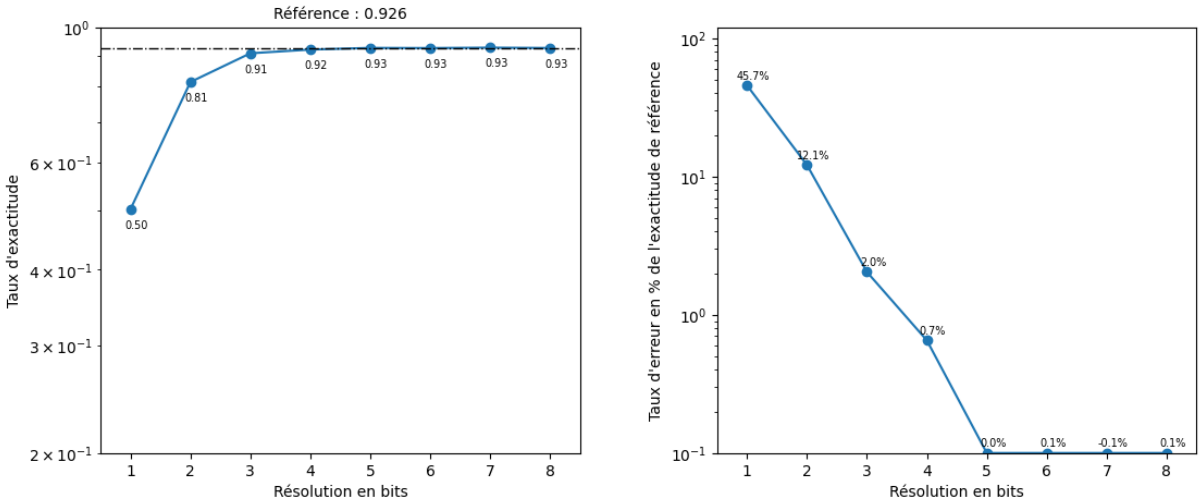

FIGURE 7.6 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution de la fonction après LDA (à gauche) et perte de performance par rapport à la référence (à droite).

A des fins de vérification, nous avons effectué une troisième fois le processus de mesure d'impact pour confirmer les résultats obtenus, en fixant la résolution des données B à 2 bits et en variant la résolution de M_A. Les mesures de performance $P_A = 92.1$, et donc notre constat d'impact sur la performance $P_R - P_A = 0.5\%$ rapporté dans la Table 7.2, étant similaire nous confirmons le choix de paramètres de résolution à 2 bits pour les données B et 4 bits pour la fonction M_A. Les performances mesurées sont rapportées dans la Table 7.2 et sur la Figure 7.7.

Pertes dues à la restriction du nombre de directions discriminantes LDA calculées

Notre protocole implique que le propriétaire des données ne peut potentiellement donner accès qu'à une part des descripteurs au concepteur de l'IA comme expliqué dans la partie II. Nous avons donc étudié la perte due à la restriction du nombre directions discriminantes LDA calculées : en appliquant les arrondis identifiés précédemment nous avons indépendamment mesuré les performances de modèles entraînés sur 1 à 4 directions discriminantes. Pour procéder à ces entraînements, les directions discriminantes sont ordonnées par leur valeur propre calculée lors de l'application de la LDA : la justification de cet ordonnancement est similaire à celle de l'ordonnancement des composantes principales de l'ACP de l'expérimentation 7.1. Les performances mesurées sont rapportées dans la

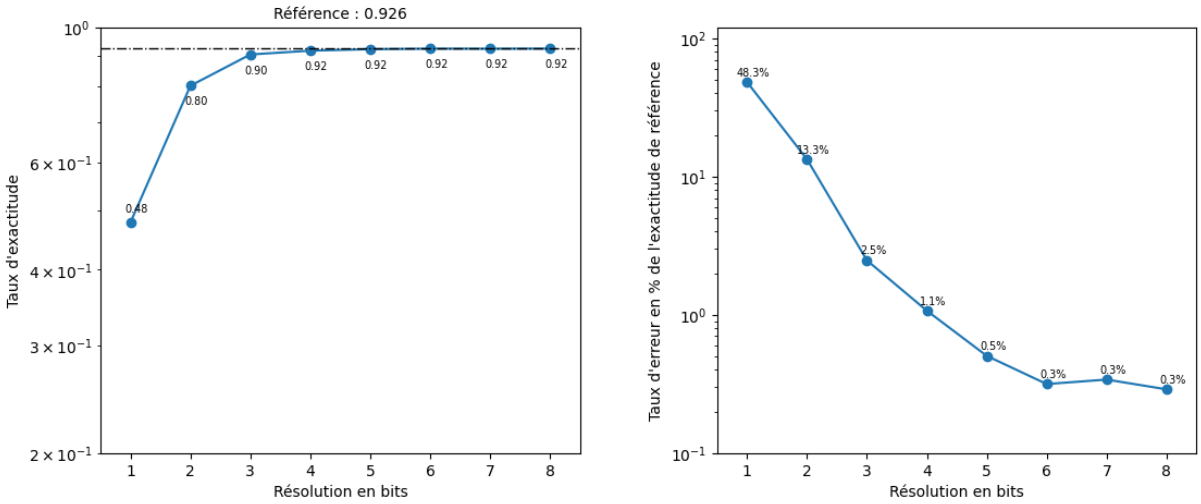

FIGURE 7.7 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution de la fonction après LDA avec une résolution fixée à 2 bits pour les données (à gauche) et perte de performance par rapport à la référence (à droite).

Table 7.2 et sur la Figure 7.8.

7.2.4 Recherche paramétrique du chiffrement fonctionnel

De la même manière que pour l'expérimentation sur l'ACP 7.1, nous déterminons les paramètres correspondants à ceux trouvés durant l'expérimentation 7.2. Pour la LDA, les résolutions en bits pour l'image et la fonction sont respectivement $B_x = 2^2$ et $B_y = 2^5$. Le modèle prend en entrée les images aplaties issues de la base de données MNIST 7, on a donc $w = 784$. Nous utilisons la recherche paramétrique décrite dans la partie 3.5. Comme pour l'expérimentation 7.1.2, la probabilité d'exactitude recherchée est $\mathbb{P}^{(1)} \geq 0.995$. En choisissant $n = 1024$, nous trouvons les paramètres suivants pour un niveau de sécurité

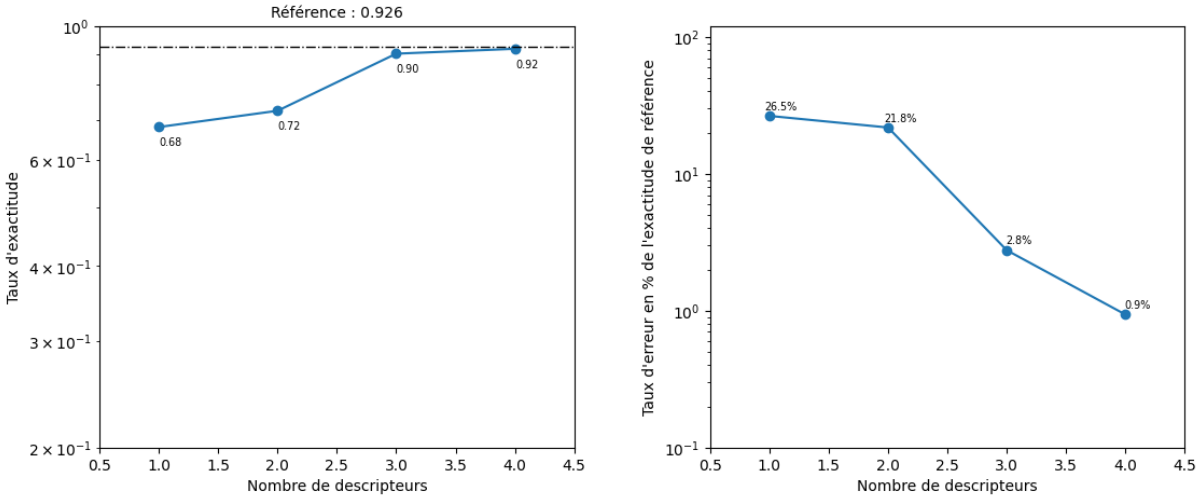

FIGURE 7.8 – Taux d'exactitude mesuré en fonction du nombre de directions discriminantes.

estimé par le *Lattice Estimator* [12] à $S_1 = 80$ bits :

$$\begin{cases} S_1 = 80.0 \text{ bits} \\ n = 1024 \\ w = 784 \\ B_x = 2^2 \\ B_y = 2^5 \\ \log(p) = 177 \\ \sigma = 202824096036516704239472512860 16 \\ \sigma_1 = \alpha_1 p = 225 \\ \sigma_2 = \alpha_2 p = 13084588083507765623896850749634641920 01 \\ \sigma_3 = \alpha_3 p = 26169176167015531247793701499269283840 00 \end{cases}$$

Et, en choisissant $n = 2048$, nous trouvons les paramètres suivants pour un niveau de sécurité estimé à $S_2 = 120$ bits :

$$\begin{cases} S_2 = 120.3 \text{ bits} \\ n = 2048 \\ w = 784 \\ B_x = 2^2 \\ B_y = 2^5 \\ \log(p) = 123 \\ \sigma = 281474976710656 \\ \sigma_1 = \alpha_1 p = 225 \\ \sigma_2 = \alpha_2 p = 36317027395115679744001 \\ \sigma_3 = \alpha_3 p = 72634054790231359488000 \end{cases}.$$

7.2.5 Architecture du modèle

L'architecture utilisée pour l'expérimentation 7.2 est la même que celle utilisée pour l'expérimentation 7.1 détaillée au paragraphe 7.2.5.

Architecture Keras			
Layer	Taille de la sortie	Paramètres	Fonction d'activation
Dense	(None, 512)	2560	ReLu
Dropout	(None, 512)	0	None
Dense	(None, 512)	26265	Relu
Dropout	(None, 512)	0	None
Dense	(None, C)	5130	Softmax

Tous les paramètres d'apprentissage sont laissés par défaut sauf :

1. Taille de batch = 128
2. Loss = categorical crossentropy
3. Optimiseur = RMSprop [110]

7.2.6 Performances des modèles entraînés suite à une LDA

Les performances mesurées sont rapportées par la Table 7.2.

TABLE 7.2 – Performance des modèles LDA

Transfert	Résolution des données	Résolution de la fonction	Descripteurs	Taux d'exactitude
Non	Pas d'arrondi	Pas d'arrondi	4	92.6% (±0.1)
Oui	Pas d'arrondi	Pas d'arrondi	4	61.2% (±0.5)
Non	0 bits	Pas d'arrondi	4	21.1% (±0)
Non	1 bits	Pas d'arrondi	4	90.2% (±0.2)
Non	2 bits	Pas d'arrondi	4	92.3% (±0.2)
Non	3 bits	Pas d'arrondi	4	92.5% (±0.2)
Non	4 bits	Pas d'arrondi	4	92.6% (±0.1)
Non	5 bits	Pas d'arrondi	4	92.6% (±0.2)
Non	6 bits	Pas d'arrondi	4	92.7% (±0.1)
Non	7 bits	Pas d'arrondi	4	92.7% (±0.1)
Non	8 bits	Pas d'arrondi	4	92.5% (±0.1)
Non	Pas d'arrondi	0 bits	4	36.5% (±0.5)
Non	Pas d'arrondi	1 bits	4	50.3% (±1.1)
Non	Pas d'arrondi	2 bits	4	81.4% (±0.4)
Non	Pas d'arrondi	3 bits	4	90.7% (±0.2)
Non	Pas d'arrondi	4 bits	4	92.0% (±0.1)
Non	Pas d'arrondi	5 bits	4	92.6% (±0.1)
Non	Pas d'arrondi	6 bits	4	92.5% (±0.2)
Non	Pas d'arrondi	7 bits	4	92.7% (±0.1)
Non	Pas d'arrondi	8 bits	4	92.5% (±0.1)
Non	2 bits	0 bits	4	34.6% (±0.3)
Non	2 bits	1 bits	4	47.9% (±0.9)
Non	2 bits	2 bits	4	80.2% (±0.2)
Non	2 bits	3 bits	4	90.3% (±0.1)
Non	2 bits	4 bits	4	91.6% (±0.3)
Non	2 bits	5 bits	4	92.1% (±0.1)
Non	2 bits	6 bits	4	92.3% (±0.1)
Non	2 bits	7 bits	4	92.3% (±0.2)
Non	2 bits	8 bits	4	92.3% (±0.2)
Non	2 bits	4 bits	1	68.0% (±0.4)
Non	2 bits	4 bits	2	72.4% (±0.3)
Non	2 bits	4 bits	3	90.0% (±0.2)
Non	2 bits	4 bits	4	91.7% (±0.1)

7.3 Application d'une couche de convolution linéaire par chiffrement fonctionnel pour la conception d'un réseau de neurones

7.3.1 Contexte et objectifs de l'expérimentation

Un CNN (Convolutional Neural Network) est un type de réseau de neurones artificiels spécialement conçu pour le traitement des images et des données en 2D. Les CNN sont inspirés de la manière dont le cerveau humain traite les informations visuelles.

Les CNN 7.3.2 sont capables d'apprendre à reconnaître des motifs dans des images grâce à des filtres de convolution successifs, extrayant ainsi des descripteurs morphologiques. En s'appuyant sur des couches de "pooling", ces descripteurs sont agrégés à mesure qu'ils sont extraits, réduisant de fait la dimension de la donnée traitée : on parle souvent d'extraction de descripteurs ou d'encodage. Ces descripteurs extraits peuvent enfin être utilisés comme entrée d'un modèle plus simple, communément un réseau de neurones, pour remplir des tâches de classification. Nous nous intéressons à son couplage au chiffrement fonctionnel.

7.3.2 Description d'une couche de convolution linéaire

Un CNN, ou Convolutional Neural Network, est un type de réseau de neurones utilisé pour le traitement d'images et d'autres types de données à structure spatiale. Il est souvent utilisé dans des tâches telles que la classification d'images, la détection d'objets et la segmentation d'images.

1. L'image d'entrée est représentée sous forme d'une matrice de pixels. Dans le cas du jeu de données MNIST, chaque pixel est représenté par un nombre entre 0 et 255, qui correspond à sa valeur sur une échelle en noir et blanc.

2. Le filtre de convolution est également représenté sous forme d'une matrice de poids. Par exemple, un filtre de taille 3×3 pourrait être représenté par la matrice :

$$[[-1, 0, 1],$$
$$[-1, 0, 1],$$
$$[-1, 0, 1]]$$

3. Par fenêtre glissante, le filtre de convolution est appliqué sur l'image d'entrée. À chaque position, une opération de produit scalaire est ainsi effectuée. L'opération de convolution, qui désigne l'ensemble de ces produits scalaires.
4. Le résultat de l'opération de convolution est stocké sous forme de matrice, communément appelée carte de caractéristiques. Intuitivement, cette carte de caractéristiques représente les régions de l'image d'entrée où le filtre a détecté certaines caractéristiques morphologiques, au sens de certaines formes.
5. Le processus de convolution est répété avec différents filtres pour extraire différentes caractéristiques de l'image d'entrée. Par exemple, un certain filtre pourrait détecter les bords, un autre filtre pourrait détecter les textures, etc.
6. Les cartes de caractéristiques obtenues à partir de différents filtres sont ensuite transformées par l'application d'une fonction d'activation pour obtenir le résultat de la couche de convolution, souvent nommées cartes de convolution.
7. Les cartes de convolution sont usuellement soumises à des couches de "pooling" avant d'être ré-exploitées en tant qu'entrées des couches suivantes du modèle. En particulier, après l'application d'une ou plusieurs couches de convolution, il est possible d'utiliser les cartes de convolution obtenues en tant que descripteurs d'entrée de couches neuronales entièrement connectées pour réaliser des tâches de détection ou de classification.

L'application du filtre de convolution pour le calcul des cartes de caractéristiques est une application linéaire calculable par produit scalaire qu'il est possible de réaliser par chiffrement fonctionnel. Le calcul des cartes de convolution, qui requièrent l'application d'une fonction d'activation souvent non linéaire, peuvent être calculées par chiffrement fonctionnel quadratique en définissant la dite fonction d'activation à l'opération de mise au carré 8.1 8.2.

7.3.3 Expérimentation et recherche de paramètres IA

A l'instar de l'ACP (cf. 7.1) et de la LDA (cf. 7.2), nous avons souhaité expérimenter le chiffrement fonctionnel pour calculer les produits scalaires nécessaires pour effectuer l'application d'une couche de convolution à des données MNIST. Ici, nous nous intéressons à l'application de schémas de chiffrement fonctionnel linéaire et donc au calcul par déchiffrement des cartes de caractéristiques d'une couche de convolution. En exploitant

la mise en forme des données décrite en Section 8.1, nous proposons une approche permettant de choisir la sécurité du schéma, tout en maîtrisant les pertes de performance induites par le protocole et l'utilisation du chiffrement fonctionnel.

Cette expérimentation s'appuie sur la même méthodologie que les expérimentations relatives à l'application de techniques de pré-traitement par chiffrement fonctionnel rapportées en Section 8.1. Mise à part la fonction effectivement appliquée, deux différences majeures sont à noter : l'architecture du modèle de machine learning est différente (cf. 7.3.5), et la méthodologie d'ordonnancement des descripteurs lors de l'étape d'estimation de l'impact de la restriction de l'information s'appuie sur des mesures de performance unitaires. Tout au long de cette expérimentation, cette architecture et les paramètres d'apprentissage utilisés ne sont pas modifiés afin d'éviter l'introduction de biais.

Nous avons donc séparé l'ensemble de données MNIST en deux parties : le jeu A contient les images étiquetées de 0 à 4 et est considéré comme la base de données publiques, et le jeu B contient les images étiquetées de 5 à 9 et est considéré comme la base de données privées cible. Un apprentissage par transfert du jeu A vers le jeu B a été mis en place, et nous avons mesuré les trois impacts sur les performances : transfert, arrondis et restriction du nombre de descripteurs.

Évaluation des performances de référence

Notre référence est construite de la même manière que dans les expérimentations précédentes : nous avons entraîné un réseau de neurones convolutifs (7.3.2) et mesuré ses performances sur la base de données B. La performance mesurée $P_R = 99.0$ est rapportée dans la Table 7.3 ; elle est considérée comme le maximum atteignable de performance de ce type de modèle sur la base de données B.

Pertes dues à l'utilisation de l'apprentissage par transfert

Ensuite, nous évaluons les performances d'un modèle par apprentissage par transfert du jeu A vers le jeu B. Pour ce faire, nous entraînons un modèle de réseau de neurones convolutionnels sur la base de données A et en extrayons la première couche de convolution, de matrice de poids K_A. Nous appliquons cette couche de convolution, sans réapprentissage, sur la base de données B et obtenons les cartes de caractéristiques des images de la base de données B. Nous entraînons ensuite un réseau de neurones correspondant à la deuxième partie de l'architecture du réseau initial (cf. 7.3.5) en utilisant les cartes

de caractéristiques obtenues comme descripteurs en entrée du modèle. Nous mesurons la performance $P_T = 98.9$ et nous évaluons la perte de performance due au transfert en calculant la différence $P_R - P_T = 0.1\%$. Cette perte, induite par la désadaptation supposée de la première couche de convolution, reste minime de par la structure similaire des jeux A et B. Les performances mesurées sont rapportées dans la Table 7.3.

Pertes dues aux arrondis du chiffrement fonctionnel

De la même manière que pour les expérimentations précédentes, nous avons ensuite étudié l'impact des arrondis de données et de fonction sur les performances du modèle. En suivant un processus identique, sont rapportées les performances mesurées pour des résolutions allant de 1 à 6 bits dans les configurations :

1. Arrondi sur les données B uniquement, en Figure 7.9 ;
2. Arrondi sur K_A uniquement, pour vérification, en Figure 7.14.
3. Arrondi sur les données B et la fonction, soit le noyau K_A pour vérification, en Figure 7.11 ;

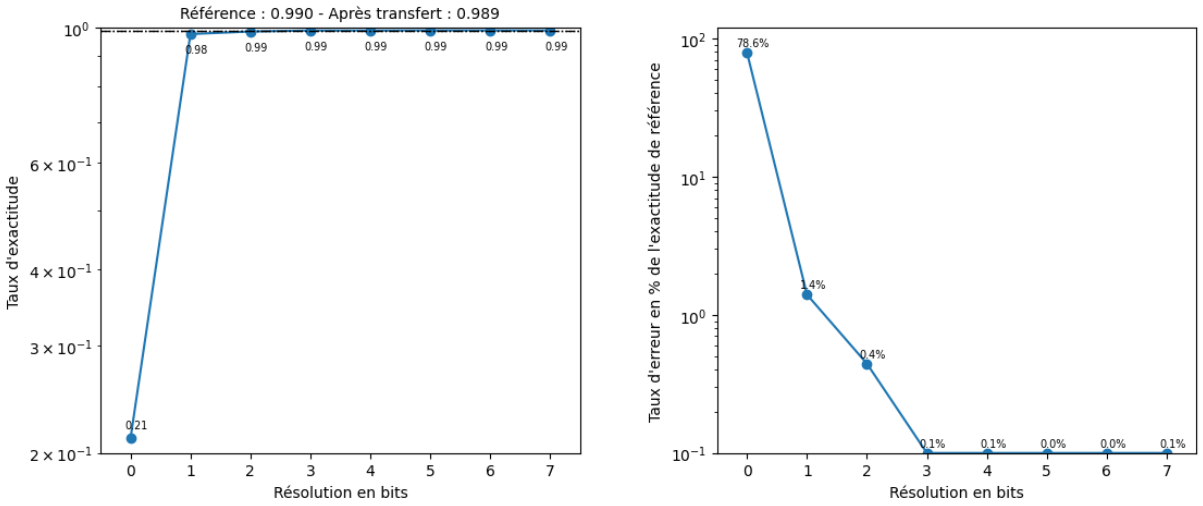

FIGURE 7.9 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution des données après une convolution linéaire (à gauche) et perte de performance par rapport à la référence (à droite).

Dans la première configuration on remarque qu'a partir de 2 bits de résolution pour les données B, l'impact sur les performances du modèle est minime. On choisit donc de fixer la résolution de l'image à 2 bits.

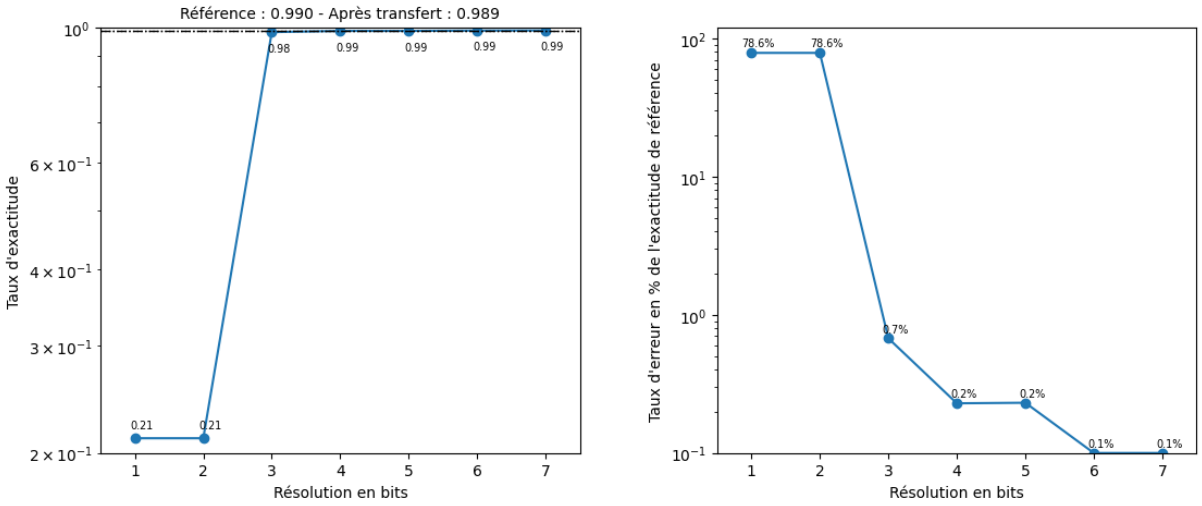

FIGURE 7.10 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution de la fonction après une convolution linéaire (à gauche) et perte de performance par rapport à la référence (à droite).

Dans la deuxième configuration, nous pouvons constater qu'avec 2 bits de résolution ou moins, le modèle présente des performances dégradées, et qu'au-delà de 3 bits de précision l'impact de l'arrondi sur les performances du modèle est négligeable. On choisit donc de fixer la précision de la fonction à 4 bits de résolutions pour ce cas d'usage.

Dans la troisième configuration, nous mesurons la performance $P_A = 98.5\%$ (cf.Table 7.3) et évaluons la perte de performance en calculant la différence $P_T - P_A = 0.4\%$. On constate qu'avec une résolution de 2 bits de précision pour les images en entrée B et de 4 bits de précision les noyaux de convolution K_A, les performances sont peu impactées. On confirme donc le choix de ces paramètres pour le restant de l'expérimentation.

Étant fixées ces valeurs de résolutions, nous identifions des paramètres pertinents pour les 3 schémas utilisés (5.2) (2.1) (3.1). Le choix de ces paramètres est expliqué en deuxième partie d'expérimentation.

Pertes dues à la restriction du nombre de cartes de caractéristiques calculées

On cherche maintenant à mesurer la perte due à la restriction du nombre de cartes de caractéristiques calculées. Pour rappel, notre protocole implique que le propriétaire des données ne peut potentiellement donner accès qu'à une part des descripteurs au concepteur de l'IA, comme expliqué dans la partie II. Le cas échéant, nous souhaitons être en mesure

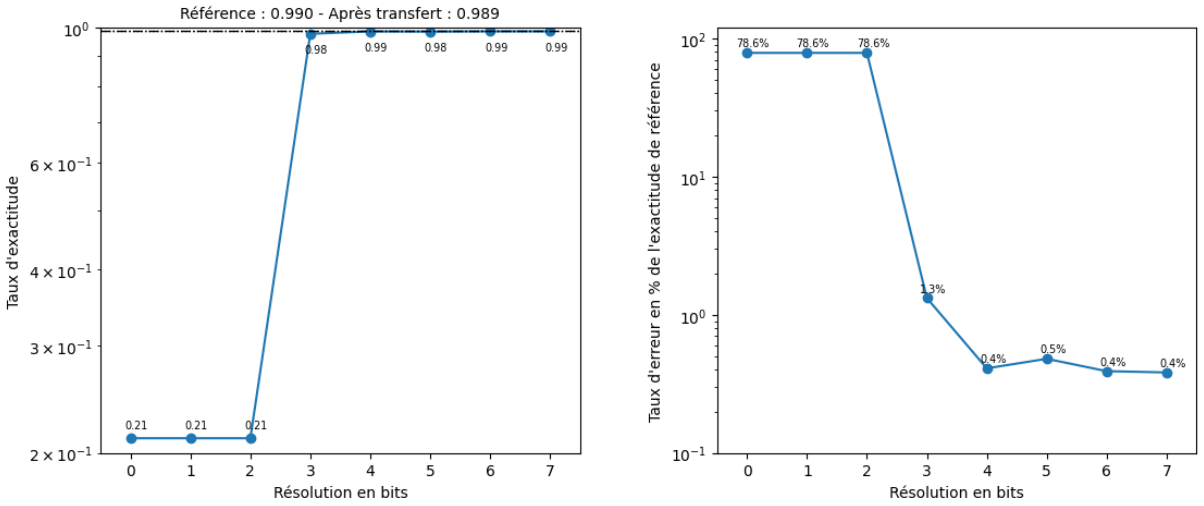

FIGURE 7.11 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution de la fonction après une convolution linéaire avec une résolution fixée à 2 bits pour les données (à gauche) et perte de performance par rapport à la référence (à droite).

d'évaluer la perte de performance qu'implique le non-accès à certains descripteurs. Pour cela, après application des arrondis induits par le chiffrement fonctionnel, nous évaluons d'abord la quantité d'information classifiante, au sens d'utile à la tâche de classification, contenue dans chaque carte caractéristique : nous entraînons indépendamment un modèle de réseaux de neurones à partir de chaque carte caractéristique.

Nous entraînons ensuite indépendamment 30 modèles de réseaux de neurones à partir de respectivement 1 à 30 descripteurs. Les descripteurs accessibles sont ici pris en compte successivement par ordre décroissant des performances associées obtenues précédemment : les descripteurs étant à priori les plus porteurs d'informations classifiantes sont ainsi ajoutés en premiers. Les performances mesurées sont rapportées dans la Table 7.4 et sur la Figure 7.12.

Nous avons choisi de les ajouter dans cette ordre car cela permet au propriétaire des données de maximiser la performance du modèle en diffusant le minimum de fonctions des données. Nous rappelons que dans un scénario d'application réelle, cette approche n'est pas directement applicable et l'évaluation de l'impact de la restriction des cartes caractéristiques accessibles devra faire l'objet d'une attention particulière.

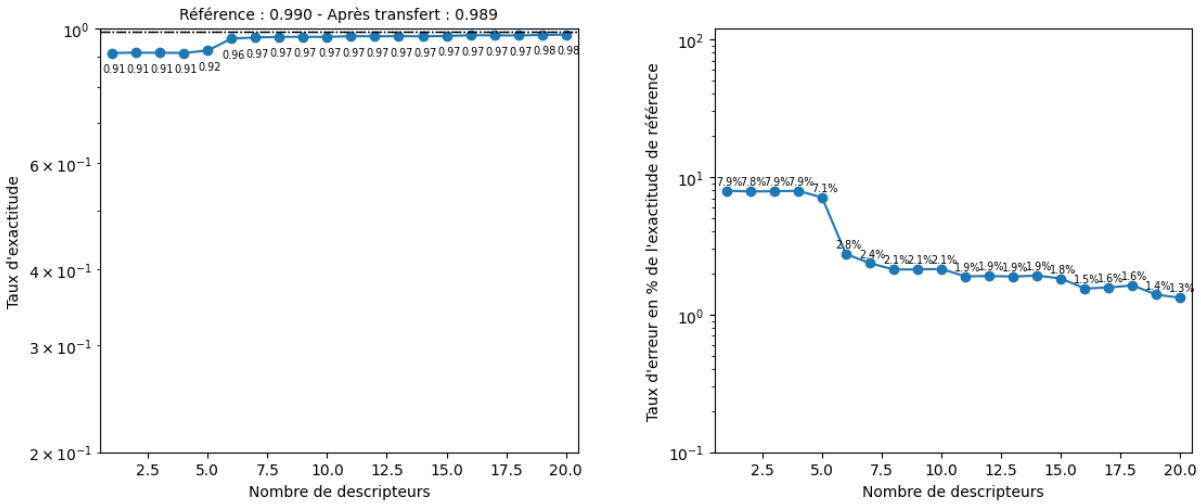

FIGURE 7.12 – Taux d'exactitude mesuré en fonction du nombre de descripteurs.

7.3.4 Recherche paramétriques du chiffrement fonctionnel

Comme pour les expérimentations précédentes (voir sections 7.1 et 7.2), nous recherchons les paramètres correspondants à ceux trouvés durant l'expérimentation 7.3. Pour cette application, on utilise le chiffrement fonctionnel pour le calcul d'un convolution linéaire de dimension 3×3. Comme décrit dans la section 8.1, la taille des entrées est de $w = 9$. De même, comme expliqué dans l'expérimentation, les résolutions choisies sont respectivement de $B_x = 2^2$ pour l'image et $B_y = 2^4$ pour la fonction. Comme pour les autres expérimentations reposant sur un chiffrement fonctionnel linéaire, nous utilisons la recherche paramétrique décrite dans la partie 3.5 et la probabilité d'exactitude recherchée est $\mathbb{P}^{(1)} \geq 0.995$. Avec $n = 1024$, nous trouvons les paramètres suivants pour un niveau

de sécurité estimé par le *Lattice Estimator* [12] à $S_1 = 80$ bits :

$$\begin{cases} S_1 = 80.0 \text{ bits} \\ n = 1024 \\ w = 9 \\ B_x = 2^2 \\ B_y = 2^3 \\ \log(p) = 79 \\ \sigma = 1073741824 \\ \sigma_1 = \alpha_1 p = 25 \\ \sigma_2 = \alpha_2 p = 824633720832001 \\ \sigma_3 = \alpha_3 p = 1649267441664000 \end{cases}$$

Et, pour $n = 2048$, nous trouvons les paramètres suivants pour un niveau de sécurité $S_2 = 122$ bits :

$$\begin{cases} S_2 = 122.6 \text{ bits} \\ n = 2048 \\ w = 9 \\ B_x = 2^2 \\ B_y = 2^3 \\ \log(p) = 56 \\ \sigma = 64 \\ \sigma_1 = \alpha_1 p = 25 \\ \sigma_2 = \alpha_2 p = 98304001 \\ \sigma_3 = \alpha_3 p = 196608000 \end{cases}.$$

7.3.5 Architecture CNN

L'architecture ci-dessous est utilisée pour l'expérimentation (7.3).

Architecture Keras				
Layer	Taille de la sortie	Paramètres	Fonction d'activation	Couche conservée après pré-apprentissage
Conv2D	(None, 26, 26, 32)	320	Linéaire	Non
MaxPooling	(None, 13, 13, 32)	0	None	Oui
Flatten	(None, 5408)	0	None	Oui
Dense	(None, 128)	692352	ReLu	Oui
Dense	(None, 5)	645	Softmax	Oui

Tous les paramètres d'apprentissage sont laissés par défaut sauf :

1. Taille de batch = 128
2. Loss = categorical crossentropy
3. Optimiseur = Adam [70]

7.3.6 Performances des modèles entraînés par réseau de convolution linéaire

Les performances mesurées sont rapportées par les Tables 7.3 et 7.4.

7.4 Application d'une Transformée de Fourier par chiffrement fonctionnel en amont de la conception d'un réseau de neurones

7.4.1 Contexte et objectifs de l'expérimentation

Comme annoncé en introduction, nous nous intéressons à la problématique de la mise à disposition sécurisée de données jugées confidentielles à des concepteurs de modèles d'intelligence artificielle. En particulier, nous considérons le cas des données de l'industrie de défense qui sont souvent considérées comme sensibles.

L'accès aux données dans l'industrie de défense par un concepteur d'IA est par essence difficile, voire impossible. Pour développer les technologies d'IA dans ce secteur, les concepteurs s'appuient donc sur des techniques palliant le manque de données réelles

TABLE 7.3 – Performance des modèles CNN partie 1 sur 2

Transfert	Résolution des données	Résolution de la fonction	Descripteurs	Taux d'exactitude
Non	Pas d'arrondi	Pas d'arrondi	30	99.0% (±0.2)
Oui	Pas d'arrondi	Pas d'arrondi	30	98.9% (±0.1)
Oui	0 bits	Pas d'arrondi	30	21.1% (±0)
Oui	1 bits	Pas d'arrondi	30	97.6% (±0.1)
Oui	2 bits	Pas d'arrondi	30	98.5% (±0.1)
Oui	3 bits	Pas d'arrondi	30	98.9% (±0.1)
Oui	4 bits	Pas d'arrondi	30	98.9% (±0.1)
Oui	5 bits	Pas d'arrondi	30	98.9% (±0.1)
Oui	6 bits	Pas d'arrondi	30	98.9% (±0.1)
Oui	7 bits	Pas d'arrondi	30	98.9% (±0.1)
Oui	Pas d'arrondi	0 bits	30	21.1% (±0)
Oui	Pas d'arrondi	1 bits	30	21.1% (±0)
Oui	Pas d'arrondi	2 bits	30	21.1% (±0)
Oui	Pas d'arrondi	3 bits	30	98.3% (±0.1)
Oui	Pas d'arrondi	4 bits	30	98.7% (±0.1)
Oui	Pas d'arrondi	5 bits	30	98.7% (±0.1)
Oui	Pas d'arrondi	6 bits	30	98.9% (±0.1)
Oui	Pas d'arrondi	7 bits	30	98.9% (±0.1)
Oui	2 bits	0 bits	30	21.1% (±0)
Oui	2 bits	1 bits	30	21.1% (±0)
Oui	2 bits	2 bits	30	21.1% (±0)
Oui	2 bits	3 bits	30	97.6% (±0.1)
Oui	2 bits	4 bits	30	98.5% (±0.1)
Oui	2 bits	5 bits	30	98.5% (±0.1)
Oui	2 bits	6 bits	30	98.6% (±0.1)
Oui	2 bits	7 bits	30	98.6% (±0.1)

TABLE 7.4 – Performance des modèles CNN partie 2 sur 2

Transfert	Résolution des données	Résolution de la fonction	Descripteurs	Taux d'exactitude
Oui	2 bits	4 bits	1	91.1% (±0.1)
Oui	2 bits	4 bits	2	91.2% (±0.2)
Oui	2 bits	4 bits	3	91.2% (±0.2)
Oui	2 bits	4 bits	4	91.1% (±0.2)
Oui	2 bits	4 bits	5	92.0% (±0.2)
Oui	2 bits	4 bits	6	96.2% (±0.1)
Oui	2 bits	4 bits	7	96.6% (±0.1)
Oui	2 bits	4 bits	8	96.8% (±0.1)
Oui	2 bits	4 bits	9	96.8% (±0.1)
Oui	2 bits	4 bits	10	96.8% (±0.1)
Oui	2 bits	4 bits	11	97.1% (±0.1)
Oui	2 bits	4 bits	12	97.1% (±0.1)
Oui	2 bits	4 bits	13	97.1% (±0.1)
Oui	2 bits	4 bits	14	97.1% (±0.1)
Oui	2 bits	4 bits	15	97.1% (±0.1)
Oui	2 bits	4 bits	16	97.4% (±0.1)
Oui	2 bits	4 bits	17	97.4% (±0.1)
Oui	2 bits	4 bits	18	97.3% (±0.1)
Oui	2 bits	4 bits	19	97.6% (±0.1)
Oui	2 bits	4 bits	20	97.6% (±0.1)

comme l'apprentissage par transfert [15, 92], l'apprentissage frugal [42, 118, 73] ou l'augmentation de données [107, 115].

Nous définissons, à des fins d'expérimentation, le scénario d'application suivant : une marine souhaite faire concevoir un algorithme de reconnaissance acoustique automatique d'un type particulier d'émission sonar exploitant les données issues des enregistrements sous-marins des sonars passifs de sa flotte.

En exploitant les schémas détaillés au paragraphe (3.1), nous cherchons donc à transformer des données issues de la base ASM (cf. 7) par produit avec une matrice de Fourier, tel que décrit au paragraphe 7.4.2.

L'objectif de cette expérience est de vérifier la faisabilité d'une application concrète d'un schéma de chiffrement fonctionnel dans le processus de conception d'un modèle d'apprentissage automatique exploitant des données réelles.

7.4.2 Description de la Transformée de Fourier

La transformée de Fourier est une technique mathématique utilisée pour analyser des signaux temporels dans le domaine fréquentiel. Elle permet de décomposer un signal en les différentes fréquences qui le composent. La transformée de Fourier discrète (DFT - Discrete Fourier Transform an anglais) est définie par la somme des produits d'un signal discret par des exponentielles complexes. La transformée de Fourier discrète est outil fondamental du Traitement du Signal, et survient dans un large panel d'applications telles que la compression audio, la détection de motifs, l'analyse spectrale, etc.

On rappelle que la transformée de Fourier $S \in \mathbb{C}^N$ d'un signal discret $s \in \mathbb{R}^N$ de $N \in \mathbb{N}^*$ échantillons est définie par

$$S = \left(\sum_{n=0}^{N-1} s(n) e^{-2i\pi kn/N} \right)_{k=0,\cdots,N-1}.$$

En définissant la matrice de Fourier $M_{\mathcal{F}} \in \mathbb{C}^{N \times N}$ telle que

$$M_{\mathcal{F}} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^1 & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)^2} \end{pmatrix}$$

avec $\omega = e^{-2i\pi/N}$, il est possible d'exprimer la transformée de Fourier discrète comme l'application linéaire

$$S = s \times M_{\mathcal{F}}$$

En machine-learning, la transformée de Fourier est souvent considérée comme un prétraitement de données permettant une représentation spectrale de données typiquement acoustiques en amont de l'entraînement d'un modèle. Nous nous intéressons à son application par chiffrement fonctionnel.

7.4.3 La métrique F1-score

Le F1-score est une métrique utilisée pour évaluer la performance d'un modèle de classification en tenant compte à la fois des faux positifs et des faux négatifs. Le F1-score est la moyenne harmonique de la précision et du rappel : la précision mesure la proportion de prédictions positives qui sont effectivement correctes, tandis que le rappel mesure la proportion de vrais positifs qui sont correctement identifiés.

Cette métrique est particulièrement utile lorsque les classes sont déséquilibrées, soit lorsque les exemples positifs et négatifs ne sont pas représentées de manière égale. Dans de tels cas, le taux d'exactitude comme seule métrique d'évaluation n'est pas adapté car un modèle peut avoir une exactitude et une précision élevées en prédisant toujours la classe majoritaire, mais un rappel très faible pour la classe minoritaire.

Mathématiquement, le F1-score se calcule de la manière suivante :

$$\begin{cases} \text{Précision} = \dfrac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{Rappel} = \dfrac{\text{TP}}{\text{TP} + \text{FN}} \\ \text{F1-score} = \dfrac{2 \times \text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} \end{cases}$$

avec TP le nombre de vrais positifs, FP le nombre de faux positifs et FN le nombre de faux négatifs.

7.4.4 Expérimentation et recherche de paramètres IA

Nous exploitons dans cette expérience un jeu de données acoustiques sous-marines (ASM), présenté au paragraphe 7, constitué de données réelles mises à disposition par

l'Université de San Francisco [61].

Nous décomposons aléatoirement ce jeu de données en un jeu d'apprentissage de 602 exemples, dont 78 exemples positifs annotés "ES", et un jeu de test de 151 exemples, dont 19 positifs.

Nous visons l'utilisation du chiffrement fonctionnel pour calculer les produits scalaires nécessaires à la transformée de Fourier : comme pour les expérimentations précédentes (7.1), (7.2) et (7.3), cette transformation linéaire est équivalente à un produit matriciel naturellement décomposable en plusieurs produits scalaires permettant chacun le calcul d'une composante fréquentielle.

Dans cette expérimentation nous proposons une approche permettant de choisir la sécurité du schéma, tout en maîtrisant les pertes de performance induites par le protocole et l'utilisation du chiffrement fonctionnel. Pour chaque cas d'utilisation, nous présentons un jeu de paramètres menant à un compromis performance/sécurité que nous jugeons pertinent.

Les données de la base considérée sont des signaux acoustiques sous-marins, notés génériquement s, de durée $D = 10$ s et de fréquence d'échantillonnage $Fe = 22050$ Hz. Nous souhaitons calculer par déchiffrements fonctionnels les composantes fréquentielles de s pour construire une représentation en temps-fréquence par la transformée de Fourier discrète sur fenêtre glissante de $N_{fft} = 1024$ points, et donc de résolution fréquentielle d'environ $21,5$Hz, avec un recouvrement de fenêtres temporelles de 75%.

Formellement, pour un signal $s \in \mathbb{R}^\ell$ et une matrice de Fourier $M_F \in \mathbb{C}^{\ell \times \ell}$ on souhaite calculer le produit matriciel $s \times M_F$. Pour cela, on considère les matrices $M_R \in \mathbb{R}^{\ell \times \ell}$ et $M_I \in \mathbb{R}^{\ell \times \ell}$ contenant respectivement les parties réelles et imaginaires de la matrice M_F tel que

$$s \times M_F = (s \times M_R) + i\,(s \times M_I)$$

Il convient de considérer ce calcul comme 2ℓ produits scalaires : ℓ produits scalaires $s \cdot M_{Ri}$ et ℓ produits scalaires $s \cdot M_{Ii}$, avec $i \in \{0, ..., \ell-1\}$. Chaque vecteur M_{Ri} et M_{Ii} correspond respectivement à une ligne des matrices M_R et M_I. La somme $M_{Ri} + iM_{Ii}$ ainsi obtenue correspond à une composante fréquentielle du signal s. Chaque bande de fréquence est considérée comme un descripteur du signal et peut être utilisée comme entrée d'un réseau de neurones.

La fonction que nous utilisons est un produit matriciel par la matrice de Fourier : cette matrice est indépendante des données et du mécanisme d'apprentissage ultérieur. Nous déroulons donc le scénario 3, décrit dans la partie II, qui adopte une configuration

où le jeu de données publiques, accessible au concepteur, et le jeu de données privées cible du propriétaire ont les mêmes classes et où la fonction calculée par FE ne varie pas en fonction des données.

L'objectif de cette expérimentation est de mesurer la perte résultante de ce processus et d'explorer une méthode de sélection de paramètres de FE permettant d'obtenir un compromis satisfaisant entre les performances du modèle de ML finalement obtenu et le niveau de sécurité du schéma de chiffrement utilisé.

En ce sens, nous évaluons différentes pertes de performance dues au protocole en fonction des paramètres choisis, par rapport à un apprentissage de référence sans restriction d'accès sur le jeu de données ASM (cf. 7). Nous identifions à ce stade que les pertes de performance peuvent être dues aux différents arrondis induits par le chiffrement fonctionnel.

L'architecture du modèle de machine-learning utilisé dans cette expérimentation est rapportée en Figure 7.4.6. Tout au long de cette expérimentation, l'architecture et les paramètres d'apprentissage ne sont pas modifiés afin d'éviter l'introduction de biais.

Évaluation des performances de référence

À des fins de référence pour comparaison, nous mesurons tout d'abord les performances d'un modèle entraîné sur des données représentées au format temps-fréquence.

Le jeu de donnée utilisé comporte des classes déséquilibré, les performances sont donc mesurées sont rapportées via la métrique du F1-score (cf. section 7.4.3). La performance $P_R = 77\%$, obtenue par moyenne des F1-score mesuré après 10 apprentissages indépendants, nous sert de référence. Elle est considérée comme le maximum atteignable de performance pour cette architecture de modèle sur la base de données ASM (7).

Pertes dues aux arrondis du chiffrement fonctionnel

On cherche maintenant à mesurer la perte due aux arrondis inhérents au FE. De fait, le schéma de chiffrement fonctionnel utilisé (3.1) s'applique dans des espaces mathématiques d'entiers et, par conséquent, les données d'entrées et la matrice de Fourier doivent être respectivement arrondies à une résolution fixée par avance. Nous procédons à l'entraînement de modèles avec la même architecture (7.4.6), en effectuant cette fois-ci le calcul de la transformée de Fourier par chiffrement fonctionnel linéaire. Le choix des paramètres correspondants au schéma (3.1), et donc à chaque modèle, est expliqué en deuxième partie

de l'expérimentation (7.4.5).

Ici, les arrondis s'appliquent aux données en entrée et à la matrice de Fourier M_F. Notez bien que plus la résolution choisie est grande, avec donc un arrondi minime, plus les paramètres du FE sont contraints, impactant négativement la sécurité ou le temps de calcul. Cependant des résolutions trop faibles, avec donc des arrondis trop importants, nuiront aux performances du modèle entraîné ensuite. On cherche donc à trouver un compromis acceptable d'arrondis minimums permettant d'obtenir des performances satisfaisantes.

Pour cela, nous étudions l'impact de l'arrondi sur les performances du modèle, en considérant des résolutions allant de 0 à 5 bits. Dans un premier temps, la performance du modèle a été mesurée en appliquant un arrondi uniquement aux données et non à la matrice de Fourier. Cette approche permet de mesurer l'effet de l'arrondi des données sur les performances du modèle. Cela implique l'application d'un arrondi aux données ASM, suivi d'un changement de base à l'aide de la matrice M_F non arrondie, et enfin l'entraînement du modèle de réseau de neurones à partir des descripteurs obtenus. Les performances mesurées sont rapportées par la Figure 7.13.

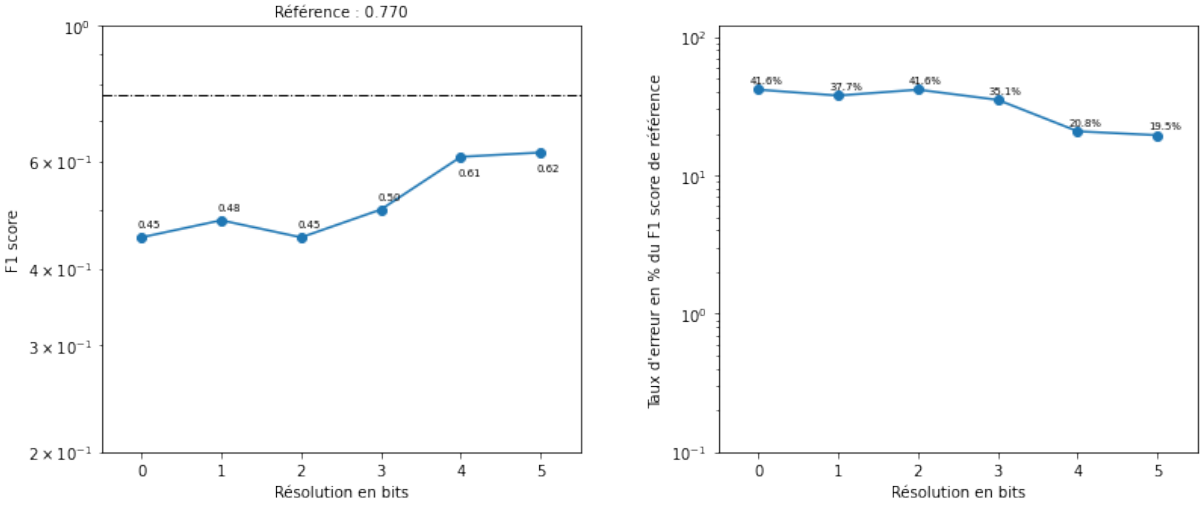

FIGURE 7.13 – F1-score mesuré en fonction du nombre de bits de résolution de la matrice de Fourier (à gauche) et perte de performance par rapport à la référence (à droite).

On remarque que les valeur de F1-score obtenue pour 4 bits et 5 bits de résolution sont significativement plus élevé que les autres. On choisit donc de fixer la résolution de l'image à 4 bits pour ce cas d'usage afin de minimiser les temps de calcul du chiffrement

fonctionnel.

Ensuite, nous répétons l'expérience en effectuant un arrondi sur la matrice de Fourier. Nous appliquons un arrondi à la matrice M_F, puis, nous entraînons le modèle de réseau de neurones en utilisant les descripteurs obtenus. Les performances mesurées sont rapportées par la Figure 7.14.

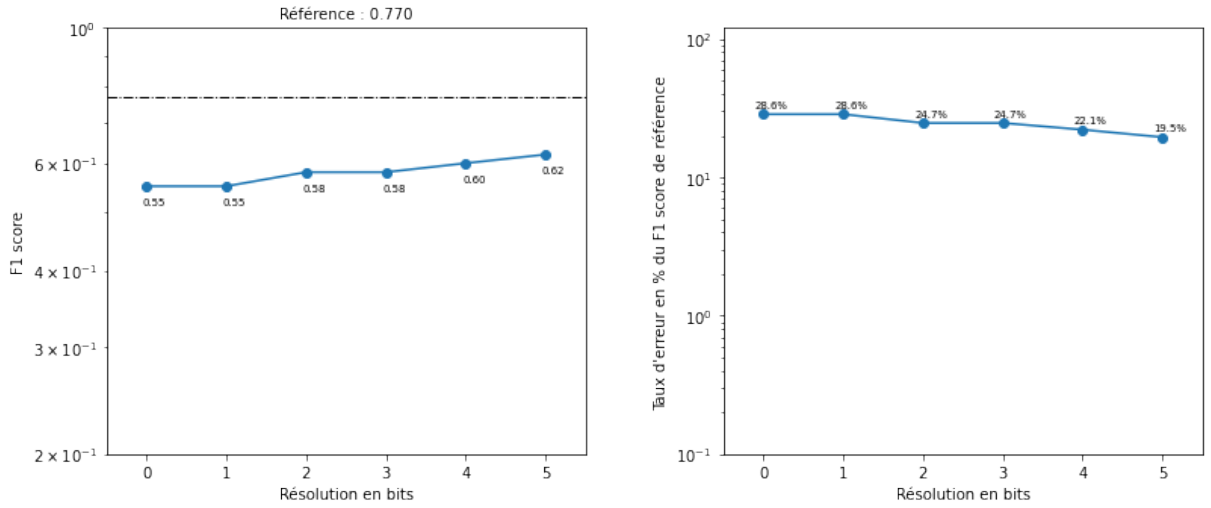

FIGURE 7.14 – F1-score mesuré en fonction du nombre de bits de résolution du signal après une Transformée de Fourier (à gauche) et perte de performance par rapport à la référence (à droite).

Nous pouvons constater qu'avec 5 bits de résolution, l'impact de l'arrondi sur les performances du modèle est minimisé. On choisit donc de fixer la précision de la matrice de Fourier à 5 bits de résolution pour ce cas d'usage.

A des fins de vérification, nous effectuons une dernière expérience sur l'arrondi pour confirmer les résultats obtenus : les valeurs de résolutions ont été choisies indépendamment, on souhaite confirmer que les pertes de performance du modèle sont négligeables une fois ces deux paramètres combinés. Pour cela, on fixe l'arrondi pour les données à 4 bits de résolution et on fait varier l'arrondi pour la matrice de Fourier. Les mesures ainsi obtenues sont rapportées par la Figure 7.15.

Nous mesurons la performance $P_A = 61\%$ et nous évaluons la perte de performance due à l'arrondi en calculant la différence $P_R - P_A = 16\%$. Cette perte de performance, bien que non-négligeable, est jugée acceptable pour la suite de l'expérimentation. On confirme donc le choix de ces paramètres (4 bits de précision pour les signaux et 5 bits de précision pour la matrice de Fourier M_F).

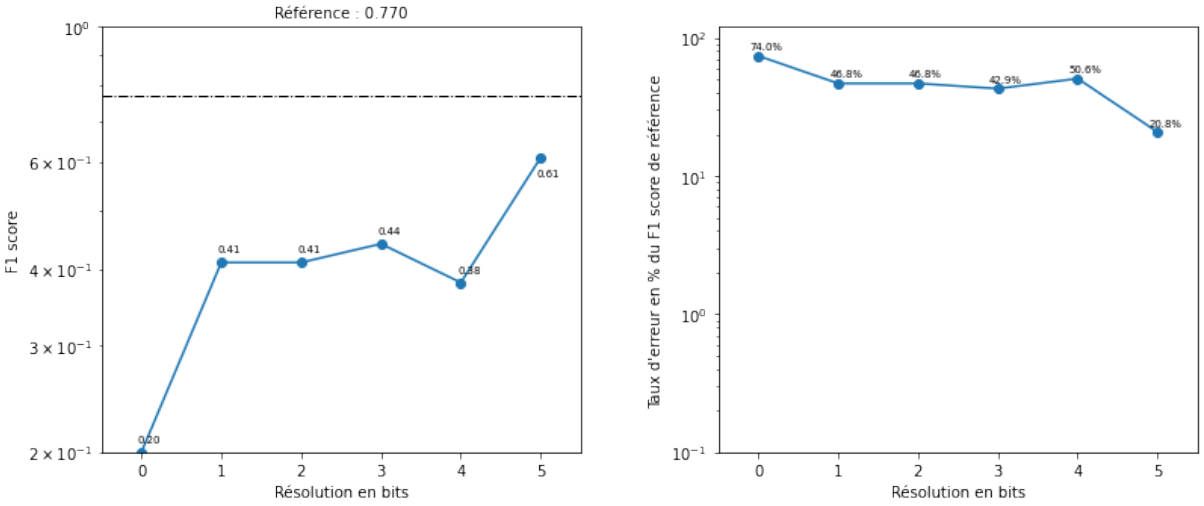

FIGURE 7.15 – F1-score mesuré en fonction du nombre de bits de résolution de la Matrice de Fourier (à gauche) et perte de performance par rapport à la référence (à droite).

7.4.5 Recherche paramétriques du chiffrement fonctionnel

Nous recherchons les paramètres correspondants à ceux trouvés durant l'expérimentation 7.4. La calcul effectué par chiffrement fonctionnel est ici un produit matriciel par la Matrice de Fourier (cf.section 7.4.2). La fenêtre temporelle glissante choisie est de taille $N_{fft} = 1024$, la taille des entrées est donc $w = 1024$. En ce basant sur les résultats obtenus lors de l'expérimentation, on choisit les tailles d'espace $B_x = 2^4$ pour le signal et $B_y = 2^5$ pour fonction équivalente à la matrice de Fourier. Comme pour les autres expérimentations reposant sur un chiffrement fonctionnel linéaire, nous utilisons la recherche paramétrique décrite dans la partie 3.5 et la probabilité d'exactitude recherchée est $\mathbb{P}^{(1)} \geq 0.995$. En choisissant $n = 1024$, nous trouvons les paramètres suivants amenant

à un niveau de sécurité estimé par le *Lattice Estimator* [12] à $S_1 = 80.3$ bits :

$$\begin{cases} S_1 = 80.3 \text{ bits} \\ n = 1024 \\ w = 1024 \\ B_x = 2^4 \\ B_y = 2^5 \\ \log(p) = 214 \\ \sigma = 217780714829400616616559748756331655331 84 \\ \sigma_1 = \alpha_1 p = 1025 \\ \sigma_2 = \alpha_2 p = 731464442511804439042371559342866257396157644800 1 \\ \sigma_3 = \alpha_3 p = 146292888502360887808474311868573251479231528960 00 \end{cases}$$

En choisissant $n = 2048$, nous trouvons les paramètres suivants pour un niveau de sécurité $S_2 = 120.0$ bits :

$$\begin{cases} S_2 = 120.0 \text{ bits} \\ n = 2048 \\ w = 1024 \\ B_x = 2^4 \\ B_y = 2^5 \\ \log(p) = 147 \\ \sigma = 36893488147419103232 \\ \sigma_1 = \alpha_1 p = 1025 \\ \sigma_2 = \alpha_2 p = 247829793020998980814766080 01 \\ \sigma_3 = \alpha_3 p = 495659586041997961629532160 00 \end{cases}$$

7.4.6 Architecture CNN-LSTM pour le Traitement du signal

L'architecture ci-dessous est utilisée pour l'expérimentation (7.4).

Architecture Keras partie 1 sur 2				
Layer	Taille de la sortie	Paramètres	Fonction d'activation	Couche conservée après pré-apprentissage
Time Distributed Conv2D	(None, None, 8, 512,8)	80	ReLu	Pas de pré apprentissage
Time Distributed Conv2D	(None, None, 4, 512, 8)	584	ReLu	Pas de pré apprentissage
Max Pooling	(None, None, 4, 256, 8)	0	None	Pas de pré apprentissage
Dropout	(None, None, 4, 256, 8)	0	None	Pas de pré apprentissage
Time Distributed Conv2D	(None, None, 4, 256, 16)	1168	ReLu	Pas de pré apprentissage
Time Distributed Conv2D	(None, None, 4, 256, 16)	2320	ReLu	Pas de pré apprentissage
Max Pooling	(None, None, 2, 128, 16)	0	None	Pas de pré apprentissage
Dropout	(None, None, 2, 128, 16)	0	None	Pas de pré apprentissage

Architecture Keras partie 2 sur 2				
Layer	Taille de la sortie	Paramètres	Fonction d'activation	Couche conservée après pré-apprentissage
Time Distributed Conv2D	(None, None, 2, 128, 32)	4640	ReLu	Pas de pré apprentissage
Time Distributed Conv2D	(None, None, 2, 128, 32)	9248	ReLu	Pas de pré apprentissage
Max Pooling	(None, None, 1, 64, 32)	0	None	Pas de pré apprentissage
Dropout	(None, None, 1, 64, 32)	0	None	Pas de pré apprentissage
Flatten	(None, None, 2048)	0	None	Pas de pré apprentissage
LSTM	(None, 32)	266368	Tanh	Pas de pré apprentissage
Dropout	(None, 32)	0	None	Pas de pré apprentissage
Dense	(None, 8)	264	ReLu	Pas de pré apprentissage
Dropout	(None, 8)	0	None	Pas de pré apprentissage
Dense	(None, 2)	18	Softmax	Pas de pré apprentissage

Tous les paramètres d'apprentissage sont laissés par défaut sauf :

1. Taille de batch = 4
2. Loss = categorical crossentropy
3. Optimiseur = Adam [70]

7.4.7 Performances des modèles entraînés par réseau de convolution LSTM

Les performances mesurées sont rapportées par la Table 7.5.

TABLE 7.5 – Performance des modèles entraînés sur donnée ASM

Transfert	Résolution des données	Résolution de la fonction	Descripteurs	F1-score
Non	Pas d'arrondi	Pas d'arrondi	1024	77.0% (±0.1)
Non	0 bits	Pas d'arrondi	1024	45% (±0.1)
Non	1 bits	Pas d'arrondi	1024	48% (±0.1)
Non	2 bits	Pas d'arrondi	1024	45% (±0)
Non	3 bits	Pas d'arrondi	1024	50% (±0)
Non	4 bits	Pas d'arrondi	1024	61% (±0.1)
Non	5 bits	Pas d'arrondi	1024	62% (±0.1)
Non	Pas d'arrondi	0 bits	1024	55% (±0.1)
Non	Pas d'arrondi	1 bits	1024	55% (±0.1)
Non	Pas d'arrondi	2 bits	1024	58% (±0.1)
Non	Pas d'arrondi	3 bits	1024	58% (±0.2)
Non	Pas d'arrondi	4 bits	1024	60% (±0.1)
Non	Pas d'arrondi	5 bits	1024	62% (±0.1)
Non	4 bits	0 bits	1024	20% (±0.2)
Non	4 bits	1 bits	1024	41% (±0.2)
Non	4 bits	2 bits	1024	41% (±0.2)
Non	4 bits	3 bits	1024	44% (±0.1)
Non	4 bits	4 bits	1024	38% (±0.2)
Non	4 bits	5 bits	1024	61% (±0.1)

Chapitre 8

Expérimentations basées sur les chiffrements fonctionnels linéaire et quadratique pour l'application d'une couche de convolution à un jeu de données

Les résultats de la section précédente montrent la faisabilité de l'utilisation du chiffrement fonctionnel linéaire pour préparer des données à des traitements de machine learning. Pour installer le chiffrement fonctionnel davantage au cœur de l'expérimentation, comme par exemple avec une fonction d'activation, il faut inclure des effets non linéaires. Le chiffrement fonctionnel quadratique est l'outil qui va nous le permettre.

8.1 Prétraitement des données

Dans cette section, nous détaillons les étapes de prétraitement de données nécessaires à l'application des schémas de chiffrement fonctionnel quadratique présentés précédemment. L'objectif ultérieur étant de calculer par déchiffrement fonctionnel des résultats d'opérations de convolutions élevées au carré, notre approche de prétraitement s'effectue en deux temps. Tout d'abord, nous transformons les données d'entrée dans un format compatible avec les espaces mathématiques des schémas, puis nous calculons la fonction g équivalente aux dites opérations. Pour ce faire, nous proposons une méthode d'encodage des données d'entrées en polynômes qui s'appuie sur l'anneau polynomial R_{p_0}, avec p_0 un nombre premier de la forme $p_0 = 2nk + 1$ majorant le résultat de l'opération quadratique à réaliser.

Lemme 8.1.1. *Soit p un nombre premier et n tel que $2n$ divise $p-1$. Si $a \in \mathbb{Z}_p^\times$ est une solution de $x^n \equiv -1 \mod p$, alors l'ensemble de toutes les solutions à cette équation est l'ensemble des puissances positives impaires de a.*

Démonstration. Soit $a \in \mathbb{Z}_p^\times$ une solution de $x^n \equiv -1 \mod p$ et k un entier. Remarquons que

$$\begin{aligned}(a^{2k+1})^n &= (a^{2k} \times a)^n \\ &= (a^n)^{2k} \times (a^n) \\ &\equiv (1) \times (-1) \qquad \mod p \\ &\equiv -1 \qquad \mod p.\end{aligned}$$

Ainsi, les puissances positives impaires de a sont effectivement des solutions de l'équation. Comme a est d'ordre $2n$, il existe exactement n valeurs distinctes de $a^{2k+1} \mod p$.

Lemme 8.1.2. *Si $p = 2nk+1$, avec n et k des entiers positifs, et $a \in \mathbb{Z}_p$ alors*

$$\begin{aligned}\frac{a}{n} &\equiv -\left(\frac{p-1}{n}\right) \times a \qquad \mod p \\ &\equiv -2ak \qquad \mod p\end{aligned}$$

Nous considérons l'opération réalisée par un filtre de convolution (voir Section 7.3.2) avec une fonction d'activation quadratique agissant sur une donnée d'entrée $X \in \mathbb{Z}^{L \times H \times D}$ et de noyau $K' \in \mathbb{Z}^{w_0 \times w_1 \times D}$. Sans application de technique de remplissage (padding en anglais), l'opération du filtre de convolution se décompose en $n = (L-w_0+1)(H-w_1+1)$ produits scalaire matriciels. Pour les évaluer par chiffrement fonctionnel, nous considérons donc les produits scalaires entre le noyau aplati $K \in \mathbb{Z}^w$, avec $w = w_0 w_1 D$, et les n sous-ensembles $X_k \in \mathbb{Z}^w, k \in \{1, ..., n\}$ de l'entrée X. Avec $L_0 = L - w_0 + 1$, le $k^{\text{ième}}$ sous-ensemble $X_k \subset X$ est défini pour tout $k \in \{1, ..., n\}$ tel que

$$X_k = (X_{a,b,c})_{1 \leq i \leq w} \qquad \text{avec} \quad \begin{cases} a = (k-1 \mod L_0) + (i-1 \mod w_0) + 1 \\ b = \left\lfloor \frac{k-1}{L_0} \right\rfloor + \left\lfloor \frac{(i-1 \mod w_0 w_1)}{w_0} \right\rfloor + 1 \\ c = \left\lfloor \frac{i-1}{w_0 w_1} \right\rfloor + 1 \end{cases}.$$

Un exemple schématique de la construction de X_k est illustré sur la Figure 8.1.

La sortie du filtre de convolution activée par un carré peut ensuite être calculée comme

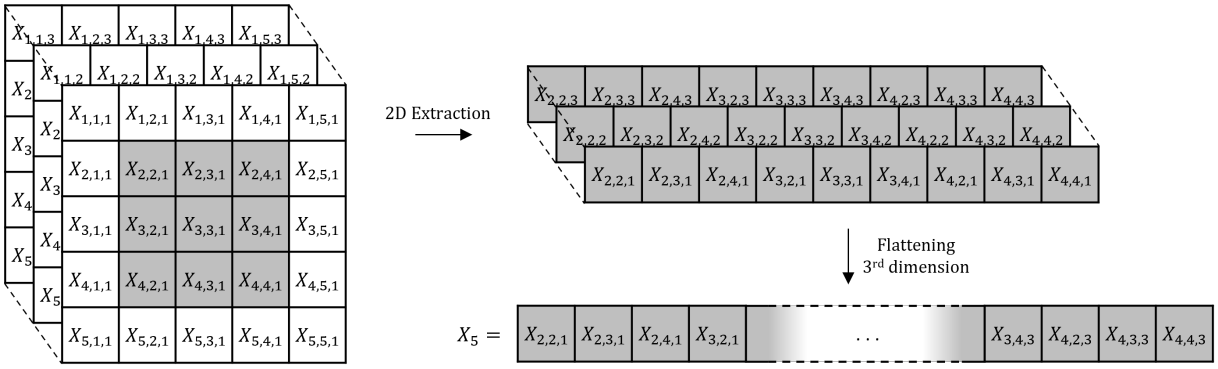

FIGURE 8.1 – Construction of X_k with $X \in \mathbb{Z}^{5\times5\times3}$, $K' \in \mathbb{Z}^{3\times3\times3}$ and $k = 4$

suit :

$$S = \left(\left(\sum_{1\leq i\leq w} K_i \left(X_k\right)_i \right)^2 \right)_{1\leq k\leq n} \in \mathbb{Z}^n.$$

Pour évaluer une telle sortie grâce au déchiffrement fonctionnel, tel que décrit dans la Section 4.2, nous définissons $g \in \mathbb{Z}^{\frac{w(w+1)}{2}}$ tel que $\forall k \in \{1, ..., n\}$ le $k^{\text{ième}}$-produit scalaire matriciel soit calculé par

$$\begin{aligned} S &= \sum_{1\leq i\leq w} (K_i(X_k)_i)^2 + 2 \sum_{1\leq j<i\leq w} K_i K_j \left(X_k\right)_i \left(X_k\right)_j \\ &= \sum_{1\leq j\leq i\leq w} g_{ij} \left(X_k\right)_i \left(X_k\right)_j \end{aligned}$$

en posant

$$g_{ij} = \begin{cases} K_i^2 & \text{quand } i = j, \\ 2K_iK_j & \text{sinon.} \end{cases}$$

Nous devons ensuite trouver un moyen d'évaluer cette combinaison quadratique de X_k en calculant une combinaison linéaire de produits polynomiaux. A ce titre, nous exploitons un nombre premier $p_0 = 2nk + 1$, avec n une puissance de 2 et $k \in \mathbb{N}^*$, tel que $p_0 > \sum_{1\leq j\leq i\leq w} g_{ij}\theta_i\theta_j \left(X_k\right)$, et recherchons maintenant le calcul dans $R_{p_0} = \mathbb{Z}_{p_0}[X]/(X^n + 1)$ de $\sum_{1\leq j\leq i\leq w} g_{ij}\theta_i\theta_j \left(X_k\right) \mod p_0$. Remarquez que dans le schéma proposé dans la Section 4.2, cette somme est bornée par K_1 par construction : p_0 pourra être choisi comme un nombre premier proche mais supérieur à K_1.

Nous cherchons donc $\theta \in R^w$ tel que

$$\forall k \in \{1, ..., n\}, \quad \sum_{1 \leq j \leq i \leq w} g_{ij} \left(X_k\right)_i \left(X_k\right)_j = \sum_{1 \leq j \leq i \leq w} g_{ij} \theta_i \theta_j \left(X_k\right) \mod p_0. \tag{8.1}$$

Notons que cela implique que θ vérifie les deux conditions suivantes :

$$\forall k \in \{1, ..., n\}, \ \forall i, j \in \{0, ..., w-1\}$$

$$\theta_i \left(A_k\right) \theta_j \left(A_k\right) = \theta_i \theta_j \left(A_k\right) \tag{8.2}$$

$$\theta_i \left(A_k\right) = \left(X_k\right)_i \tag{8.3}$$

pour des $A_k \in \mathbb{Z}_{p_0}$ à trouver également.

La condition (8.2) est vérifiée si $\forall k \in \{1, ..., n\}$, tous les A_k sont des racines n-ièmes distinctes de -1, c'est-à-dire

$$A_k{}^n \equiv -1 \mod p_0.$$

Le polynôme $X^n + 1$ est un polynôme unitaire avec n racines. Pour identifier ces A_k, on pourra par exemple appliquer l'algorithme de Berlekamp [19]. En pratique, nous en trouvons un premier, noté A_0, en utilisant une approche par force brute, puis nous en déduisons les autres A_k en utilisant le Lemme 8.1.1. À partir des n valeurs de A_k distinctes et des X_k, on peut calculer $\theta \in R^w$ par interpolation telle que la seconde condition (8.3) soit vérifiée. Cette interpolation peut être réalisée assez rapidement grâce à une adaptation de la transformée de Fourier discrète inverse et du Lemme 8.1.2 : si les A_k sont définis tels que $\forall k \in \{1, ..., n\}, A_k = {A_0}^{2k+1}$, on peut définir une matrice de transformation appropriée $\mathcal{M} \in \mathbb{Z}_{p_0}^{n \times n}$ par

$$\begin{aligned} \forall i, j \in \{1, ..., n\} \qquad \mathcal{M}_{i,j} &\equiv -\left(\frac{p_0 - 1}{n}\right) \times \left({A_0}^{-(ij \bmod n)}\right)^2 \times {A_0}^j \mod p_0 \\ &\equiv -\left(\frac{p_0 - 1}{n}\right) \times \left({A_0}^{j - 2 \times (ij \bmod n)}\right) \mod p_0 \end{aligned}$$

et la matrice de coefficient de θ peut finalement être calculée par le produit $X'^{\intercal} \times \mathcal{M}$, avec $X' = \left(X_k\right)_{0 \leq k \leq n-1} \in \mathbb{Z}_{p_0}^{n \times w}$. Avec les A_k et θ vérifiant (8.2) et (8.3), il nous est possible de calculer par déchiffrement fonctionnel le polynôme $\sum_{1 \leq j \leq i \leq w} g_{ij} \theta_i \theta_j$ qui peut ensuite être

évalué pour chaque A_k pour obtenir

$$\forall k \in \{1, ..., n\}, \quad \sum_{1 \leq j \leq i \leq w} g_{ij} \theta_i \theta_j \left(A_k\right) = \sum_{1 \leq j \leq i \leq w} g_{ij} \left(X_k\right)_i \left(X_k\right)_j \mod p_0$$

qui est le résultat recherché.

8.2 Application d'une couche de convolution quadratique par chiffrement fonctionnel pour la conception d'un réseau de neurones

8.2.1 Contexte et objectifs de l'expérimentation

Le modèle utilisé dans cette expérimentation est un réseau de neurones à convolution (CNN - Convolutional Neural Network en anglais) [72] comme introduit dans l'expérimentation 7.3 à la différence que les fonctions d'activation du réseau de neurones sont quadratiques et non pas linéaires comme dans la précédente experimentation.

Rappelons qu'un réseau de neurones à convolution (CNN) est un type de réseau de neurones spécialement conçu pour le traitement des images et des données en 2D. Inspiré par la façon dont le cerveau humain traite les informations visuelles, un CNN est utilisé pour des tâches telles que la classification d'images, la détection d'objets et la segmentation d'images. Les CNN sont capables d'apprendre à reconnaître des motifs dans les images en utilisant des filtres de convolution successifs, ce qui permet d'extraire des descripteurs morphologiques des images. Ces descripteurs sont ensuite agrégés à l'aide de couches de sous-échantillonnage, ce qui réduit la dimension des données traitées. Les descripteurs extraits peuvent ensuite être utilisés comme entrée pour des modèles plus simples, comme un réseau de neurones, afin de remplir des tâches de classification.

En exploitant le schéma détaillé précédemment dans la partie 4.2 , nous cherchons donc à transformer des données issues de la base MNIST (cf. 7) par convolution quadratique.

L'objectif de cette expérience est de vérifier la faisabilité d'une application concrète d'un schéma de chiffrement fonctionnel quadratique basé sur le RLWE dans le processus de conception d'un modèle d'apprentissage automatique.

8.2.2 Description d'une couche de convolution quadratique

Comme expliqué dans la section 7.3.2, Le filtre de convolution est l'une des principales composantes d'un CNN. Ce filtre s'appuie sur une matrice de poids, ou noyau (kernel en anglais), appliquée par convolutions successives à la donnée d'entrée pour en extraire des caractéristiques importantes. En pratique, l'application d'une couche de convolutions est assimilée au processus suivant :

1. Une donnée d'entrée est considérée sous forme d'une matrice de valeurs numériques. Dans le domaine de traitement d'image, on parle de matrice de pixels, chacun étant encodé en valeurs entières ; une image en noir et blanc est usuellement décrite par des pixels de valeurs comprises entre 0 et 255 par exemple.

2. Le filtre de convolution est également représenté sous forme d'une matrice de valeurs de poids. Par exemple, un filtre de noyau de taille 3×3 pourrait ressembler à :

$$[[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]].$$

3. La convolution de la donnée d'entrée par le filtre défini s'effectue par somme pondérée glissante. En considérant successivement chaque sous-ensemble de l'image d'entrée de même dimension que le noyau du filtre, on calcule le résultat de l'opération de convolution par produit scalaire du dit sous-ensemble.

4. Les résultats de l'opération de convolution sont stockés en tant que matrice de valeurs, souvent appelée carte de convolution. On interprète cette carte de convolution comme une représentation des régions de la donnée d'entrée où le filtre a détecté certaines caractéristiques morphologiques.

5. Le processus de convolution est répété avec chacun des filtres de la couche de convolution, pour extraire différentes caractéristiques de la donnée d'entrée. Par exemple dans le domaine de traitement d'image, un filtre pourrait détecter des bords de formes au sein d'une image, un autre filtre pourrait détecter les textures, etc.

6. Les cartes de caractéristiques obtenues à partir de différents filtres sont ensuite utilisées comme entrée pour les couches suivantes du CNN, telles que les couches de pooling et les couches entièrement connectées, pour effectuer des opérations supplémentaires de classification ou de détection d'objets.

L'application du filtre convolution pour le calcul des cartes de caractéristiques est une application linéaire calculable par produit scalaire qu'il est possible de réaliser par chiffrement fonctionnel. Le calcul des cartes de convolutions, qui requièrent l'application

d'une fonction d'activation souvent non linéaire, peuvent être calculées par chiffrement fonctionnel quadratique en définissant la dite fonction d'activation à l'opération de mise au carré (cf. 8.1 et 8.2).

On s'intéresse dans cette expérimentation à l'application d'un filtre de convolution suivit de cartes de convolutions à fonctions d'activation quadratiques. L'ensemble de cette application est calculable par une somme polynomiale quadratique de la forme

$$\sum_{1 \leq j \leq i \leq w} g_{ij} \theta_i \theta_j,$$

avec $\theta \in R^w$ un vecteur de polynôme et $g \in \mathbb{Z}^{\frac{w(w+1)}{2}}$ un vecteur scalaire (cf. 8.1). C'est cette fonction que nous calculons par chiffrement fonctionnel quadratique (cf. 3).

8.2.3 Expérimentation et recherche de paramètres IA

Nous utilisons le chiffrement fonctionnel pour calculer les opérations nécessaires pour effectuer l'application d'une couche de convolution quadratique à des données issues de la base de données MNIST (cf. 7). On cherche à calculer l'ensemble des cartes d'activation (cf. 7.3.2 et 8.1) en sortie d'une couche de convolution activée par mise au carré par FE : chaque carte d'activation découle d'un filtre de convolution distinct, et nous cherchons l'application dudit filtre par déchiffrement fonctionnel utilisant une unique clé fonctionnelle.

Dans cette expérimentation nous proposons une approche permettant de choisir la sécurité du schéma, tout en maîtrisant les pertes de performance induites par le protocole et l'utilisation du chiffrement fonctionnel. Nous présentons des jeux de paramètres pertinents pour chaque cas d'utilisation, c'est-à-dire des paramètres permettant d'obtenir un compromis performance/sécurité optimisé.

La méthode utilisée pour l'expérimentation implique de réduire les données MNIST par produit de convolution. Chacune des 8 cartes caractéristiques est calculée par 11×11 fonctions quadratique pour chaque image (cf. 8.2.4).

Comme les expérimentations 7.1 et 7.3, nous nous reposons sur le scénario 1 décrit dans la section II. Tout d'abord, nous divisons l'ensemble de données MNIST en deux parties : un premier ensemble de données A qui contient les images étiquetées de 0 à 4 et est considéré comme la base de données publiques, et un deuxième ensemble de données B qui contient les images étiquetées de 5 à 9 et est considéré comme la base de données

privées cible. Comme expliqué dans le scénario 1, nous effectuons un apprentissage par transfert sécurisé en utilisant le chiffrement fonctionnel du jeu A vers le jeu B. L'objectif de cette expérimentation est de mesurer la perte résultante de ce processus et d'explorer une méthode de sélection de paramètres de chiffrement fonctionnel permettant d'obtenir un compromis satisfaisant entre les performances du modèle de ML obtenu à la fin et le niveau de sécurité du schéma de chiffrement utilisé.

Nous examinons les différentes pertes de performance causées par le protocole, en fonction des paramètres sélectionnés, par rapport à un apprentissage standard sans restriction d'accès sur le jeu de données B. À ce stade, nous classons les pertes de performance en trois catégories distinctes :

1. Les pertes dues à l'utilisation de l'apprentissage par transfert
2. Les pertes dues à l'arrondi du chiffrement fonctionnel
3. Les pertes dues à la restriction du nombre de cartes de convolutions calculées

Nous étudions ces trois types de pertes successivement.

En premier lieu, nous étudions les pertes dues à l'utilisation de l'apprentissage par transfert en comparant les performances d'un modèle entraîné sans contrainte sur le jeu de données B avec celle d'un modèle entraîné par transfert. Ensuite nous effectuons des tests afin de déterminer des paramètres pertinents en cherchant les minimums de résolutions de données et de fonction amenant à un minimum de pertes de performance du modèle. Enfin, nous étudions méthodiquement les pertes dues à la restriction du nombre de cartes d'activation calculées, dans un objectif de minimiser le nombre de fonctions partagées par le propriétaire des données.

L'architecture du modèle utilisé est décrite dans la Figure 8.2.4. Tout au long de cette expérimentation, cette architecture et les paramètres d'apprentissage utilisés ne sont pas modifiés afin d'éviter l'introduction de biais.

A des fins de référence de comparaison, nous mesurons tout d'abord les performances d'un modèle de machine learning entraîné sans contraintes, composé d'un réseau de neurones convolutionnels quadratiques (cf. 8.2.4 pour l'architecture), sur la base de données B. Les performances mesurées sont rapportées via la métrique d'exactitude, accuracy en anglais, dans la Table 8.1. La performance $P_R = 98.8\%$ obtenue nous sert de référence, et est considérée comme le maximum atteignable de performance de ce type de modèle sur la base de données B.

Ensuite, nous évaluons les performances d'un modèle ayant la même architecture mais conçu à l'aide de l'apprentissage par transfert. Pour ce faire, nous entraînons un modèle

de réseau de neurones convolutionnels à fonction d'activation quadratique, toujours avec l'architecture décrite partie 8.2.4, sur la base de données A. Puis nous extrayons les poids de la première couche de convolution du modèle obtenue après apprentissage. Autrement dit nous extrayons les poids K_A des noyaux de convolution de la première couche. Nous appliquons cette couche de convolution, sans réapprentissage, sur la base de données B et obtenons les cartes de convolutions des images de la base de données B. Nous entraînons ensuite un réseau de neurones correspondant à la deuxième partie de l'architecture du réseau initial (cf. 8.2.4) en utilisant les cartes de convolutions obtenues comme descripteurs en entrée du modèle. Nous mesurons la performance $P_T = 98.6\%$ et nous évaluons la perte de performance due au transfert en calculant la différence $P_R - P_T = 0.2\%$. Cette perte s'explique par la désadaptation induite par le fait que la matrice de passage M_A utilisée n'a pas été calculée sur le jeu de données cible B. Cependant, étant donné que le jeu de données publiques d'entraînement présente une structure similaire, elle est minime.

On cherche maintenant à mesurer la perte due à l'arrondi inhérent au FE. De fait, le schéma de chiffrement fonctionnel utilisé (4.2) s'applique dans des espaces mathématiques d'entiers et par conséquent, les données d'entrées et la fonction doivent être respectivement arrondies à une résolution fixée par avance. Nous procédons à l'entraînement de modèles avec la même architecture (8.2.4), en effectuant cette fois-ci la convolution par chiffrement fonctionnel linéaire. Le choix des paramètres du schéma est expliqué en deuxième partie de l'expérimentation 8.2.6.

Ici, les arrondis se portent donc sur les données B en entrée et sur les noyaux K_A. Notez bien que plus la résolution choisie est grande, avec donc un arrondi minime, plus les paramètres du FE sont contraints, impactant négativement la sécurité ou le temps de calcul. Cependant des résolutions trop faibles, avec donc des arrondis trop importants, nuiront aux performances du modèle entraîné ensuite. On cherche donc à trouver un compromis acceptable d'arrondis minimums permettant d'obtenir des performances satisfaisantes.

Pour cela, nous étudions l'impact de l'arrondi sur les performances du modèle, en considérant des résolutions allant de 1 à 7 bits. Dans un premier temps, la performance du modèle a été mesurée en appliquant un arrondi uniquement aux données et non à la fonction. Cette approche permet de mesurer l'effet de l'arrondi des données sur les performances du modèle. Cela implique l'application d'un arrondi aux données B, suivi d'une convolution à l'aide des noyaux K_A non arrondie, et enfin l'entraînement du modèle de réseau de neurones à partir des descripteurs obtenus. Les performances mesurées sont rapportées par la Figure 8.2.

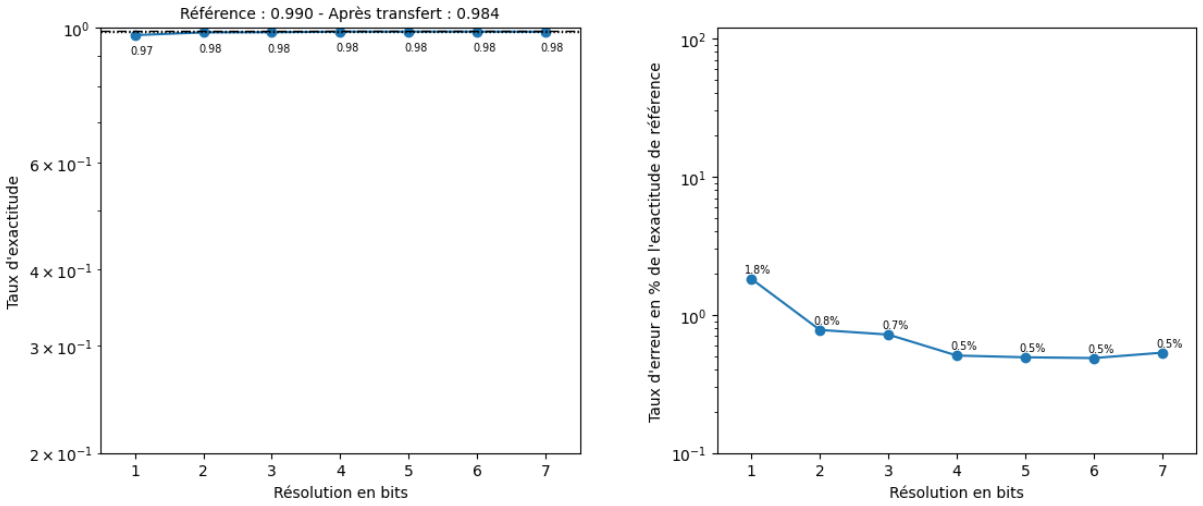

FIGURE 8.2 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution des données après une convolution quadratique (à gauche) et perte de performance par rapport à la référence (à droite).

On remarque qu'à partir de 1 bits de résolution, l'impact d'un plus grand arrondi sur les performances du modèle est minime et qu'à partir de 2 bits l'impact sur les performances du modèle est minime. On choisit donc de fixer la résolution de l'image à 2 bits pour ce cas d'usage.

Ensuite, nous répétons l'expérience en effectuant un arrondi sur la fonction. Nous appliquons un arrondi à la matrice M_A, puis nous effectuons le changement de base sur les données B non arrondies. Enfin, nous entraînons le modèle de réseau de neurones en utilisant les descripteurs obtenus. Les performances mesurées sont rapportées dans la Table 8.1 et sur la Figure 8.3.

Nous pouvons constater qu'avec 2 bits de résolution ou moins, le modèle présente des performances dégradées, et qu'à partir de 3 bits de précision, l'impact de l'arrondi sur les performances du modèle est négligeable. On choisit donc de fixer la précision de la fonction à 4 bits de résolution pour ce cas d'usage.

A des fins de vérification, nous effectuons une dernière expérience sur l'arrondi pour confirmer les résultats obtenus : les valeurs de résolutions ont été choisies indépendamment, on souhaite confirmer que les pertes de performance du modèle sont négligeables une fois ces deux paramètres combinés. Pour cela, on fixe l'arrondi pour les données à 2 bits de résolution et on fait varier l'arrondi pour la fonction. Cela signifie que l'on applique un arrondi à la matrice M_A puis on effectue le changement de base sur les données B ar-

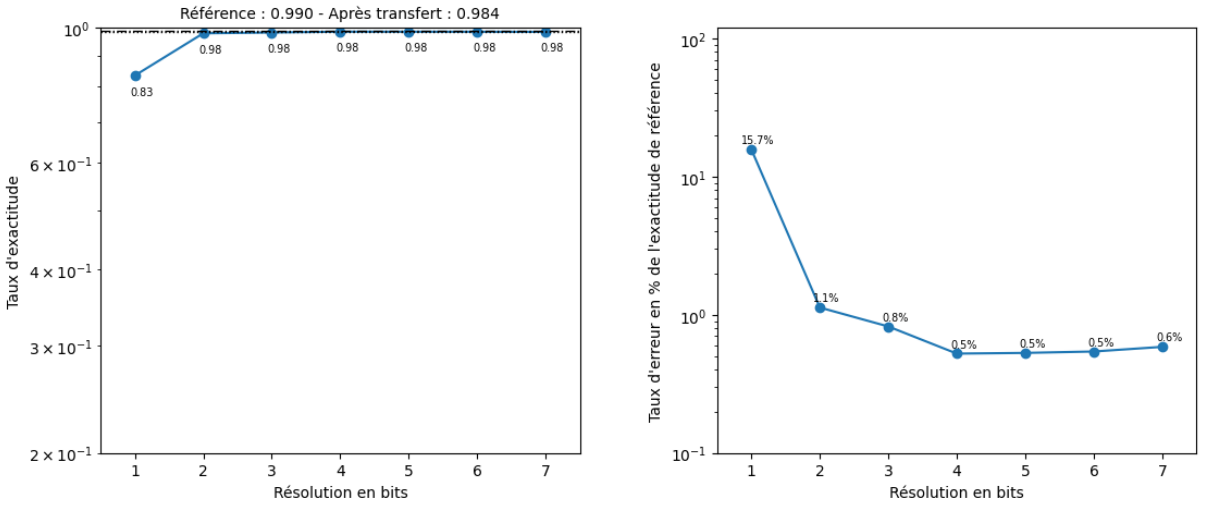

FIGURE 8.3 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution de la fonction après une convolution quadratique (à gauche) et perte de performance par rapport à la référence (à droite).

rondies à 2 bits de précision. Les performances mesurées sont rapportées dans la Table 8.1 et sur la Figure 8.4.

Nous mesurons la performance $P_A = 98.1$ et nous évaluons la perte de performance due à l'arrondi en calculant la différence $P_T - P_A = 0.5\%$. On constate qu'avec résolution de 2 bits de précision pour les images en entrée B et de 3 bits de précision les noyaux de convolution K_A, les performances sont peu impactées. On confirme donc le choix de ces paramètres pour le restant de l'expérimentation.

On cherche maintenant à mesurer la perte due à la restriction du nombre de cartes de convolutions calculées. En effet notre protocole implique que le propriétaire des données ne peut potentiellement donner accès qu'à une part des descripteurs au concepteur de l'IA comme expliqué dans la partie II. Le cas échéant, il faut être en mesure d'évaluer la perte de performance qu'implique le non-accès à certains descripteurs. Pour cela, nous appliquons les mêmes transformations qu'au cours de l'expérimentation précédente sur la base de données B cible : nous effectuons un apprentissage par transfert et un arrondi à une résolution de 2 bits pour les images et à une résolution de 3 bits pour la matrice de passage. Ensuite, dans un premier temps nous évaluons la quantité d'information classifiante contenue dans chaque cartes caractéristiques. Pour cela, nous entraînons ensuite indépendamment 8 modèles de réseaux de neurones à partir de chaque descripteurs. L'architecture est la même que pour la première partie de l'expérimentation post-transfert

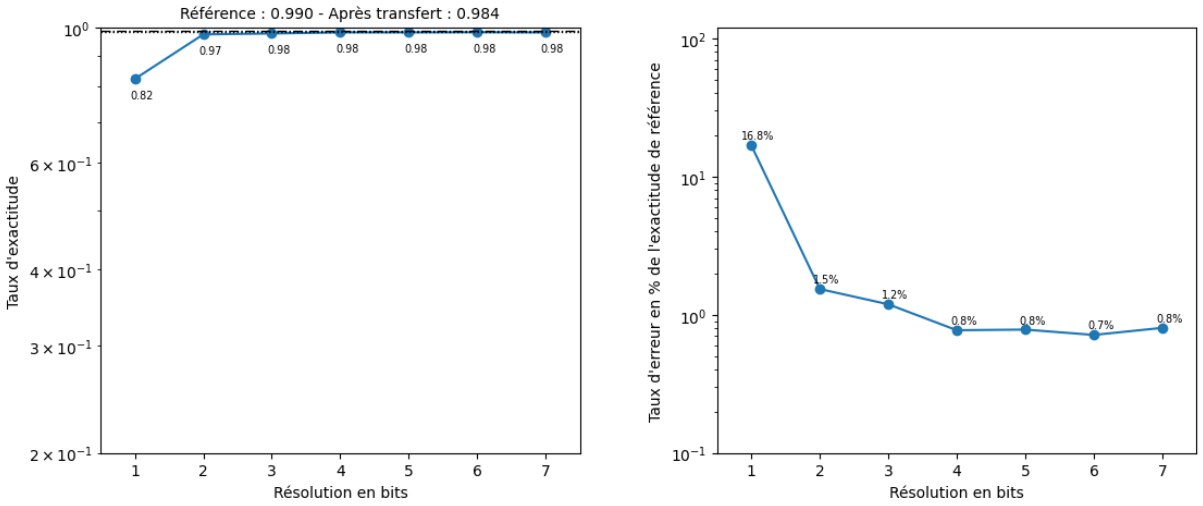

FIGURE 8.4 – Taux d'exactitude mesuré en fonction du nombre de bits de résolution de la fonction après une convolution quadratique avec une résolution fixée à 2 bits pour les données (à gauche) et perte de performance par rapport à la référence (à droite).

(cf. 8.2.4), seule la taille de l'entrée est de 1 au lieu de 8.

Nous entraînons ensuite indépendamment 8 modèles de réseaux de neurones à partir de respectivement 1 à 8 descripteurs. Les modèles ont tous l'architecture décrite Figure 8.2.4, seule la taille de l'entrée varie selon le nombre de descripteurs sélectionnés. Les descripteurs accessibles sont ici ajoutés par ordre croissant d'exactitude contenue. Les descripteurs étant à priori les plus porteurs d'informations classifiantes, au sens d'utile à la classification, sont ainsi ajoutés en premiers. Les performances mesurées sont rapportées dans la Table 8.1 et sur la Figure 8.5.

Nous avons choisi de les ajouter dans cette ordre car cela permet au propriétaire des données de maximiser la performance du modèle en diffusant le minimum de fonctions des données. Dans un scénario d'application réelle, il n'y a pas nécessairement de corrélation entre la sensibilité de la fonction calculée et la part d'information classifiante qu'elle contient : le propriétaire des données communique au concepteur de l'IA les fonctions auxquelles il souhaite lui donner accès, sans garantie de variance totale expliquée. En retour, le concepteur de l'IA peut estimer les éventuelles pertes de performance dues à la restriction d'information. Le détail de ce protocole est inclus dans la partie II.

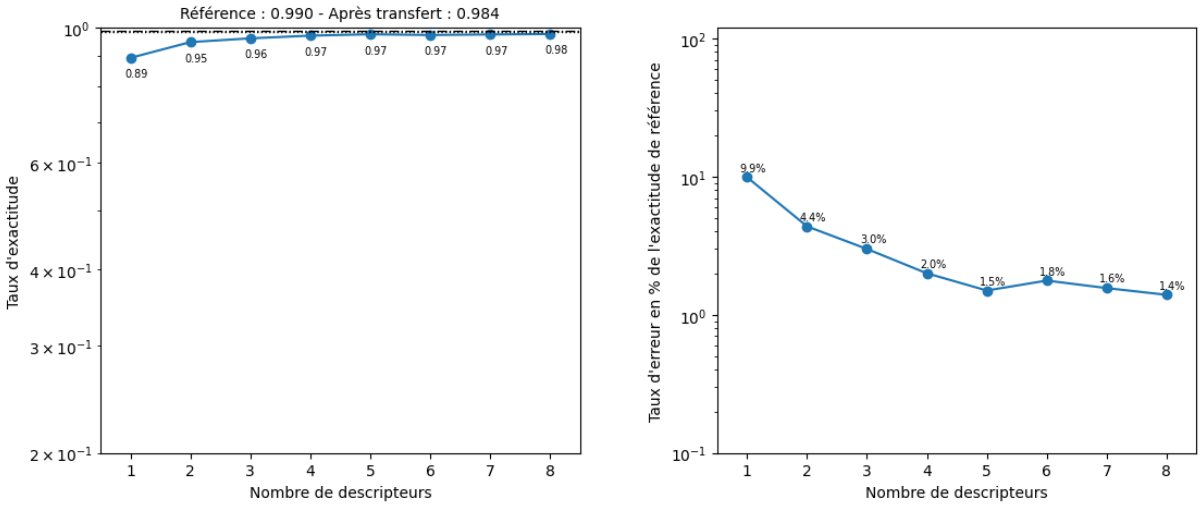

FIGURE 8.5 – Taux d'exactitude mesuré en fonction du nombre de descripteurs.

8.2.4 Architecture CNN quadratique

Nous utilisons l'architecture ci dessous pour l'expérimentation 8.2.

Architecture Keras				
Layer	Taille de la sortie	Paramètres	Fonction d'activation	Couche conservée après pré-apprentissage
Conv2D	(None, 26, 26, 4)	40	ReLu	Non
MaxPooling	(None, 13, 13, 4)	0	None	Non
Conv2D	(None, 11, 11, 8)	288	Square	Non
MaxPooling	(None, 5, 5, 8)	0	None	Oui
Flatten	(None, 200)	0	None	Oui
Dense	(None, 128)	25728	ReLu	Oui
Dense	(None, 5)	645	Softmax	Oui

Tous les paramètres d'apprentissage sont laissés par défaut sauf :

1. Taille de batch = 128
2. Loss = categorical crossentropy
3. Optimiseur = Adam [70]

8.2.5 Performances des modèles entraînés par réseau de convolution quadratique

Les performances mesurées sont rapportées par la Table 8.1.

8.2.6 Recherche paramétriques du chiffrement fonctionnel quadratique

A partir des résultats obtenus lors de l'expérimentation, nous recherchons les paramètres du chiffrement fonctionnel quadratiques adéquats. Le protocole suivi pour cette recherche paramétrique est expliqué Section 4.6. Le chiffrement fonctionnel est utilisé pour le calcul d'un convolution quadratique de dimension 3×3 , la taille des entrées est donc $w = 9$. De plus, les résolutions choisies ici sont respectivement $B = 2^2$ et $\|g\|_\infty = 2^3$ (voir Section 8.2.3. La probabilité d'exactitude recherchée est $\mathbb{P}^{(1)} \geq 0.995$. En choisissant $n = 2048$, nous trouvons les paramètres suivants pour des niveaux de sécurité estimés par le *Lattice Estimator* [12] à $S_q = 243$ bits pour le chiffrement quadratique et $S = 70.1$ bits pour le sur-chiffrement linéaire :
Pour le chiffrement fonctionnel quadratique :

$$\begin{cases} \text{Sécurité} = 243 \text{ bits} \\ n = 2048 \\ w = 9 \\ B = 2^2 \\ \|g\|_\infty = 2^3 \\ \log(p_1) = 26 \\ \sigma = 1073741824 \\ \sigma_1 = \alpha_1 p_2 = 25 \\ \sigma_2 = \alpha_2 p_2 = 824633720832001 \\ \sigma_3 = \alpha_3 p_2 = 1649267441664000 \end{cases}$$

Pour le sur-chiffrement linéaire :

$$\begin{cases} S = 70.1 \text{ bits} \\ n = 2048 \\ w = 9 \\ B_x = p_1 \\ B_y = p_1 \\ \log(p_2) = 439 \\ \sigma = 64 \\ \sigma_1 = \alpha_1 p_2 = 25 \\ \sigma_2 = \alpha_2 p_2 = 98304001 \\ \sigma_3 = \alpha_3 p_2 = 196608000 \end{cases}.$$

TABLE 8.1 – Performance des modèles CNN

Transfert	Résolution des données	Résolution de la fonction	Descripteurs	Taux d'exactitude
Non	Pas d'arrondi	Pas d'arrondi	8	98.8% (±0)
Oui	Pas d'arrondi	Pas d'arrondi	8	98.6% (±0)
Oui	0 bits	Pas d'arrondi	8	21.1% (±0)
Oui	1 bits	Pas d'arrondi	8	97.2% (±0.2)
Oui	2 bits	Pas d'arrondi	8	98.2% (±0.1)
Oui	3 bits	Pas d'arrondi	8	98.2% (±0.2)
Oui	4 bits	Pas d'arrondi	8	98.5% (±0.2)
Oui	5 bits	Pas d'arrondi	8	98.5% (±0.1)
Oui	6 bits	Pas d'arrondi	8	98.5% (±0.1)
Oui	7 bits	Pas d'arrondi	8	98.4% (±0.1)
Oui	Pas d'arrondi	0 bits	8	21.1% (±0)
Oui	Pas d'arrondi	1 bits	8	83.4% (±0.3)
Oui	Pas d'arrondi	2 bits	8	97.8% (±0.1)
Oui	Pas d'arrondi	3 bits	8	98.1% (±0.1)
Oui	Pas d'arrondi	4 bits	8	98.4% (±0)
Oui	Pas d'arrondi	5 bits	8	98.4% (±0.1)
Oui	Pas d'arrondi	6 bits	8	98.4% (±0)
Oui	Pas d'arrondi	7 bits	8	98.4% (±0.1)
Oui	2 bits	0 bits	8	21.1% (±0)
Oui	2 bits	1 bits	8	82.3% (±0.2)
Oui	2 bits	2 bits	8	97.4% (±0.1)
Oui	2 bits	3 bits	8	97.8% (±0.2)
Oui	2 bits	4 bits	8	98.1% (±0)
Oui	2 bits	5 bits	8	98.2% (±0)
Oui	2 bits	6 bits	8	98.2% (±0.1)
Oui	2 bits	7 bits	8	98.2% (±0.1)
Oui	2 bits	4 bits	1	89.2% (±0.4)
Oui	2 bits	4 bits	2	94.6% (±0.1)
Oui	2 bits	4 bits	3	96.0% (±0.1)
Oui	2 bits	4 bits	4	97.0% (±0.1)
Oui	2 bits	4 bits	5	97.5% (±0.2)
Oui	2 bits	4 bits	6	97.2% (±0.1)
Oui	2 bits	4 bits	7	97.4% (±0.1)
Oui	2 bits	4 bits	8	97.6% (±0.1)

Chapitre 9

Conclusion

9.1 Bilan des travaux

Dans ce manuscrit de thèse nous avons rapporté un état de l'art des différentes approches du PPML. Nous avons porté une attention particulière aux techniques de cryptographie émergentes comme le chiffrement fonctionnel et les schémas de chiffrement basés sur les réseaux. Nous avons notamment étudié les schémas de FE basés sur le LWE linéaire [8] [85] et quadratique [9] en détail. En particulier, nous avons proposé des méthodes afin de calculer les bornes et les probabilités d'exactitude de chaque schéma. Nous avons aussi proposé des méthodes afin d'évaluer la sécurité de ces différents schémas, et notamment celle du schéma proposé par [9].

Nous avons proposé des approches utilisant des schémas de chiffrement fonctionnel pour la conception de modèle d'apprentissage automatique à partir de données considérées confidentielles. Pour cela nous avons proposé différents scénarios couvrant des situations différentes que l'on peut retrouver dans le monde industriel, comme celui du naval de défense, en adressant divers types de pré-traitement applicables sur les données (cf. sections 7.1, 7.2 7.3, 7.4, et 8.2) . Ces scénarios (cf. II) s'appuient pour la plupart sur la technique d'apprentissage par transfert [72], et nous avons rapporté une méthodologie d'évaluation d'impact sur les performances du modèle et quelques résultats expérimentaux sur deux bases de données [75] et [61].

Ces propositions méthodologiques ont fait l'objet d'expérimentations, dans lesquelles nous démontrons l'applicabilité du chiffrement fonctionnel pour le calcul de prétraitements communs comme l'analyse par composante principale, l'analyse discriminante linéaire ou la transformée de Fourier, mais aussi pour l'application de couches neuronales de convolutions linéaires et quadratiques. Au titre de ces expérimentations, nous avons proposé une méthode de recherche de paramètres minimisant les pertes de performances tout en limitant au maximum l'information partagée par le propriétaire.

9.2 Réflexions personnelles sur les limites et opportunités d'application du chiffrement fonctionnel

Dans l'ensemble mon constat est que l'utilisation des méthodes de cryptographie émergentes n'est pas une réponse immédiate à la problématique de l'apprentissage de modèles IA. En particulier dans le cas du chiffrement fonctionnel, les approches utilisées reposent sur des hypothèses fortes en lien avec la capacité du propriétaire des données à estimer si la fonction calculée, ou l'information déchiffrée, compromettent ses données. Les attaques par inférences, caractérisées par l'inférence d'informations autres que celles a priori autorisées, reste un problème latent complexe à adresser. Une des perspectives ouvertes quant à la recherche pour l'application du FE est donc la proposition d'une méthode permettant au propriétaire d'évaluer si la fonction partagée par chiffrement fonctionnel à l'utilisateur compromet la confidentialité de ses données. Il semble à première vue que ce type de méthode dépendra fortement de la définition donnée à la confidentialité des données, et donc du cas d'application visé. Cet aspect est selon moi un facteur limitant majeur pour l'application concrète du FE dans l'industrie. Des approches reposant sur l'explicabilité de l'IA [87, 111] et notamment de visualisation des données sont possibles : nous avons envisagé de proposer ce type de méthodes avec notamment des approches basées sur des auto-encodeurs variationnels [71] ou convolutionnels [84], permettant au propriétaire de visualiser ce que l'utilisateur est capable de reconstituer comme information à partir des fonctions partagées. Ce type d'approches ne fonctionne cependant efficacement que pour des données visualisables, et ne répond pas à la problématique de manière formelle parce que s'appuyant finalement sur une appréciation humaine. Toute la difficulté de cette problématique repose sur le fait que chaque cas d'usage peux avoir des contraintes très spécifiques selon le type de données traitées et selon la part d'information confidentielle contenue.

Pour le moment l'approche se heurte à une antinomie entre les principes fondamentaux de la cryptographie et de l'IA. D'une part, les schémas de chiffrement sont conçus pour ne laisser filtrer aucune information, les chiffrés devant avoir une distribution indistinguable de celle d'un oracle aléatoire. D'autre part, la valeur ajoutée des approches par IA par contraste aux approches métiers est la capacité d'identifier et d'exploiter des informations clés au sein d'un ensemble complexe de nombreuses variables. Le compromis à trouver entre le partage d'information suffisante pour entraîner une IA performante tout en protégeant les données repose lui aussi sur l'hypothèse forte que la partie de l'information

classifiante des données n'est pas sensible. Cela peut être le cas dans certains cas d'usage mais pose une limite rédhibitoire pour certains autres.

Des cas d'usage propices à l'utilisation du FE pour la conception ou l'utilisation de modèles de machine learning existent cependant. En particulier, l'application de modèles pré-entraînés sur des données sensibles fait l'objet de plus en plus de recherche, et semble naturelle depuis l'avènement des larges modèles, en traitement du langage par exemple. Un des scénarios crédibles, que j'ai pu identifier dans l'industrie, est rencontré lorsqu'un client souhaite faire classifier ses données sans les partager au propriétaire du modèle. Ce scénario survient régulièrement, que ce soit pour des raisons législatives, de défense, ou même lorsqu'un maître d'oeuvre cherche à mettre en concurrence plusieurs concepteurs afin d'évaluer le modèle le plus performant ou simplement avant engagement financier. Dans tous les cas, l'idéal serait que la fonction calculable par FE soit équivalente à l'application du modèle entier : le déchiffrement fonctionnel donnerait accès au résultat souhaité directement, limitant de fait les risques de compromission d'informations et simplifiant les procédures apparentes de calcul. Ce cas de figure existe déjà pour certains modèles, en particulier ceux reposant sur des fonctions linéaires et quadratiques mais on envisage qu'à terme, avec par exemple l'extension des fonctionnalités du FE à des fonctions non-linéaires plus complexes, il sera possible de déchiffrer directement la sortie d'un modèle neuronal. A cette fin, il serait intéressant d'étudier les techniques de réduction de modèle d'IA pré-entrainé pour "simplifier" les fonctions visées. A titre d'exemple, on pourrait s'intéresser aux techniques de relinéarisation de modèles qui permettrait de calculer des prédictions complexes par l'application de schémas de chiffrement fonctionnel linéaire.

Au cours de mes travaux, j'ai eu l'occasion d'étudier des techniques de réduction de dimension simples mais il en existe beaucoup d'autres. Selon le cas d'usage, les pré-traitements ou les réductions peuvent être très efficaces. Je pense que le développement du chiffrement fonctionnel passera par la définition de méthodes de réduction de dimensions plus rigoureuses qui permettront une évaluation plus fine du type et de la quantité d'information contenue après transformation. Conjointement avec un hypothétique élargissement du périmètre de fonctions calculables par FE, l'incorporation d'un modèle entier au sein d'une clé fonctionnelle ne semble pas si lointaine.

Par extension, les schémas de chiffrement fonctionnel à fonction cachée [35, 69, 112], sont d'intérêt : même si le modèle entier était "contenu" dans une clé fonctionnelle, sa divulgation pourrait être considéré comme une compromission d'informations. En effet à partir des poids d'un modèle neuronal par exemple, un attaquant extérieur est théo-

riquement en mesure d'inférer des informations sur les données d'apprentissage utilisées du concepteur. La possibilité de réaliser un schéma de chiffrement fonctionnel avec fonction cachée permettrait la définition de scénarios applicatifs tendant vers une sécurité des informations maximum.

Un autre cas d'usage du FE que j'ai pu identifier dans l'industrie du naval de défense est son utilisation à bord des systèmes embarqués comme les navires de surfaces, les sous-marins ou, de manière plus probante, les drônes. Le FE permettrait de stocker à bord les données sensibles récoltées en ne donnant l'accès aux différents sous-systèmes et équipements qu'à la part d'information utile à leur mission. Par la même, les données enregistrées au cours de la mission peuvent être stockées au format chiffré et seraient protégées en cas de capture ou de compromission ultérieure.

On pourra également penser à l'utilisation du FE pour le partage d'information entre industriels. Plutôt que d'avoir à calculer une fonction des données sur l'ensemble des données historiques récoltées à chaque partage d'information, un industriel pourrait chiffrer et partager ses données à mesure qu'elles sont générées et donner accès ponctuellement aux nouvelles fonctions à partager par une simple génération de clé fonctionnelle. On notera que dans cette perspective, le coût de chiffrement est constant et les efforts de calcul, soit ici de déchiffrement, sont délégués aux demandeurs.

Par ailleurs, la maturité du FE pour des cas d'usages industriels dépend des méthodes de sélection des paramètres de schémas, en fonction des cas d'usages et du niveau de sécurité souhaité. Cette problématique de la sélection de paramètres s'apparente à un problème d'optimisation multicritère où l'on souhaite minimiser le temps de calcul tout en maximisant la sécurité, sous contrainte de probabilité d'exactitude. Dans nos travaux nous avons proposé une méthode itérative simple qu'il reste à améliorer. Les contributions à l'état de l'art proposant des schémas théoriques proposent en général des preuves de sécurité abstraites d'un haut niveau mathématiques : la transition vers une estimation concrète de sécurité comparable aux sécurités des schémas publiquement utilisés n'est pas toujours immédiate. Bien sûr, le niveau de sécurité exact ne peut être évalué fermement car les capacités des attaquants évoluent chaque jour et en ce sens le Lattice Estimator [12], régulièrement mis à jour par la communauté scientifique, est un atout précieux.

Également à noter, je me suis intéressé lors de mes travaux aux différents problèmes difficiles garantissant la sécurité des constructions cryptographiques en anticipation de l'ère post-quantique (cf. section 1.8). Même les spécialistes du domaine ne peuvent dire avec précision quand les technologies quantiques seront en état de fonctionner efficace-

ment pour les attaques sur les cryptosystèmes. Ce notamment à cause des techniques de "Store Now, Decrypt Later" adoptées aujourd'hui par les attaquants, il semble important pour l'ensemble des acteurs, en particulier étatiques, financiers et de santé, d'anticiper ce moment. Les institutions en charge de l'anticipation de l'ère post-quantique indiquent majoritairement que les problèmes basés sur les réseaux et donc en particulier le LWE et ses variantes sont la piste principale pour contrer les attaques post-quantiques, alors que les cryptosystèmes basés sur les problèmes du logarithme discret ou du RSA sont menacés voir même identifiés comme obsolètes.

Annexe A

Annexe : environnement de travail

Sauf mention contraire, les expérimentations rapportées dans ce document ont été implémentées et exécutées sur un ordinateur portable avec les spécifications suivantes :

— Système d'exploitation : Windows 11
— Processeur : Intel Core i7-10750H 2.60GHz
— Mémoire : 16Go DDR4 2933MHz
— Processeur graphique : NVIDIA RTX 2060 Max-Q Design 6Go

Encore, sauf mention contraire, les expérimentations rapportées ont été implémentées et exécutées dans un environnement logiciel Python en version 3.9.2. Pour l'implémentation des éléments de machine learning, les paquets scikit-learn 1.2.2 [63] et tensorflow 2.13.0 [65] sont utilisés. Les éléments graphiques ont été générés en utilisant le paquet matplotlib 3.7.2 [62]. Le paquet librosa 0.10.0.post2 [60] a été utilisé pour l'ouverture et le traitement des fichiers acoustiques de l'expérimentation rapportée en partie 7.4.

Par ailleurs, afin d'implémenter les schémas de chiffrement (5.2) (2.1) (3.1) (4.2), nous avons utilisé le logiciel SageMath en version 9.2 [64], vu comme une extension dédiée au calcul symbolique de Python. Ce logiciel nous a notamment permis d'implémenter facilement les espaces mathématiques utilisés comme les anneaux polynomiaux modulés symétriques. Également, SageMath permet la déclaration de variables à précision arbitraire et a facilité les expérimentations des schémas basés sur le DDH (5.2) (5.1) ainsi que l'évaluation de la sécurité des schémas (2.1) (3.1) (4.2).

A.1 SageMath

SageMath est un logiciel libre de calcul basé sur Python qui permet de résoudre des problèmes mathématiques complexes, utilisable en tant que paquet Python ou au travers d'une interface interactive. Il peut être utilisé pour effectuer des calculs numériques et symboliques, pour tracer des graphiques, pour implémenter des algorithmes mathématiques ou pour effectuer des simulations. Dans notre cas, l'avantage de SageMath est qu'il

permet de gérer de nombreux objets mathématiques différents, y compris les anneaux polynomiaux modulés, et les opérations associées : arithmétique, factorisation, dérivation... En particulier, SageMath offre des fonctionnalités dédiées aux anneaux polynomiaux modulaires, comme la possibilité de calculer le polynôme minimal d'un élément donné ou de trouver les racines d'un polynôme.

SageMath offre également une meilleure gestion des précisions de valeurs numériques, au sens de la résolution en nombre de bits, que Python. Pour cela, SageMath s'appuie sur la bibliothèque GMP (GNU Multiple Precision Arithmetic Library) qui permet de gérer efficacement les nombres de grande taille avec une précision arbitraire. GMP est implémentée en langage C et fournit des fonctions pour effectuer des opérations arithmétiques sur des nombres entiers, des nombres rationnels et des nombres à virgule flottante avec une précision arbitraire.

A.2 Python

Python est un langage de programmation interprété, orienté objet, de haut niveau et multi-plateforme. Il a été créé en 1991 par Guido van Rossum et est devenu l'un des langages de programmation les plus populaires au monde. Python est connu pour sa simplicité, sa lisibilité et sa facilité d'apprentissage. Il est utilisé dans de nombreux domaines tels que la science des données, l'apprentissage automatique, la création de sites web, la robotique, etc. Python dispose d'une grande bibliothèque standard qui facilite la réalisation de tâches courantes. Il est également extensible, ce qui signifie que des modules tiers, ou paquets, peuvent être ajoutés pour étendre ses fonctionnalités.

En particulier pour le machine learning, Python a de nombreux avantages :

1. Bibliothèques spécialisées : Python dispose de plusieurs bibliothèques dédiées de machine learning telles que Scikit-learn, TensorFlow, Keras, PyTorch, etc. Ces bibliothèques fournissent des implémentations d'algorithmes et outils utiles aux tâches d'entraînement, d'évaluation de performances et d'utilisation de modèles en inférence.

2. Grande communauté : De par sa popularité, Python profite d'une grande communauté de développeurs travaillant sur des projets de machine learning et partageant leurs connaissances et leurs expériences.

3. Intégration avec d'autres technologies : Il est aujourd'hui possible d'interfacer Python avec d'autres technologies, en particulier du domaine de gestion de données massives telles que Hadoop, Spark ou SQL. Ces interfaces permettent la conception de modèles né-

cessitant de grandes quantités de données.

A.3 Scikit-learn

Scikit-learn est une bibliothèque open-source de Machine Learning pour le langage de programmation Python. Elle propose des outils simples et efficaces pour la classification, la régression, le clustering et la réduction de dimension entre autres.

En particulier, Scikit-learn offre des implémentations d'algorithmes utilisés dans nos expérimentations comme l'ACP ou la LDA, incluant des outils de visualisation de résultats utiles en phase de développement.

A.4 Tensorflow

TensorFlow est une bibliothèque open-source de calcul numérique et de machine learning, développée par Google. Elle permet de créer des modèles de réseaux de neurones, y compris des réseaux de neurones convolutifs (CNN), que nous utilisons dans nos expérimentations.

TensorFlow est réputée flexible et performante et peut être utilisée pour créer une grande variété de modèles de réseaux de neurones, y compris des modèles de réseaux de neurones profonds.

Parce que très populaire et largement utilisée dans la communauté de l'apprentissage automatique, il est facile de trouver des ressources, tutoriels et exemples de code Tensorflow.

A.5 Keras

Keras est une bibliothèque open-source de Deep Learning écrite en Python. Elle est conçue pour simplifier et accélérer l'implémentation de modèles d'apprentissage profond, notamment en offrant des fonctionnalités usuelles comme les couches de neurones convolutifs (CNN) par exemple.

Aujourd'hui inclue dans les distributions Tensorflow, Keras est compatible avec plusieurs autres bibliothèques d'apprentissage profond, dont notamment Theano et CNTK. Au même titre que Tensorflow, Keras profite d'une grande popularité et d'une communauté particulièrement active.

BIBLIOGRAPHIE

[1] Michel Abdalla, Fabrice Benhamouda et Romain Gay, « From single-input to multi-client inner-product functional encryption », in : *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2019, p. 552-582.

[2] Michel Abdalla et al., « Decentralizing inner-product functional encryption », in : *IACR International Workshop on Public Key Cryptography*, Springer, 2019, p. 128-157.

[3] Michel Abdalla et al., « Simple functional encryption schemes for inner products », in : *IACR International Workshop on Public Key Cryptography*, Springer, 2015, p. 733-751.

[4] Abbas Acar et al., « A survey on homomorphic encryption schemes : Theory and implementation », in : *ACM Computing Surveys (Csur)* 51.*4* (2018), p. 1-35.

[5] Shweta Agrawal, Dan Boneh et Xavier Boyen, « Efficient lattice (H) IBE in the standard model », in : *Advances in Cryptology–EUROCRYPT 2010 : 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, Springer, 2010, p. 553-572.

[6] Shweta Agrawal, Dan Boneh et Xavier Boyen, « Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE », in : *Advances in Cryptology–CRYPTO 2010 : 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*, Springer, 2010, p. 98-115.

[7] Shweta Agrawal, Rishab Goyal et Junichi Tomida, « Multi-input quadratic functional encryption from pairings », in : *Advances in Cryptology–CRYPTO 2021 : 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part IV 41*, Springer, 2021, p. 208-238.

[8] Shweta Agrawal, Benoıt Libert et Damien Stehlé, « Fully secure functional encryption for inner products, from standard assumptions », in : *Annual International Cryptology Conference*, Springer, 2016, p. 333-362.

[9] Shweta Agrawal et Alon Rosen, « Functional encryption for bounded collusions, revisited », in : *TCC* (2017), p. 173-205.

[10] Shweta Agrawal et al., « Adaptive simulation security for inner product functional encryption », in : *IACR International Conference on Public-Key Cryptography*, Springer, 2020, p. 34-64.

[11] Miklós Ajtai, « Generating hard instances of lattice problems », in : *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, p. 99-108.

[12] Martin R Albrecht, Rachel Player et Sam Scott, « On the concrete hardness of learning with errors », in : *Journal of Mathematical Cryptology* 9.*3* (2015), p. 169-203.

[13] Joël Alwen et al., « On the relationship between functional encryption, obfuscation, and fully homomorphic encryption », in : *Cryptography and Coding : 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings 14*, Springer, 2013, p. 65-84.

[14] Prabhanjan Ananth et Vinod Vaikuntanathan, « Optimal bounded-collusion secure functional encryption », in : *Theory of Cryptography Conference*, Springer, 2019, p. 174-198.

[15] Eva Artusi et Fabien Chaillan, « Automatic recognition of underwater acoustic signature for naval applications », in : *1st Maritime Situational Awareness Workshop MSAW 2019*, 2019.

[16] Nuttapong Attrapadung et Benoıt Libert, « Functional encryption for inner product : Achieving constant-size ciphertexts with adaptive security or support for negation », in : *Public Key Cryptography–PKC 2010 : 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings 13*, Springer, 2010, p. 384-402.

[17] Carmen Elisabetta Zaira Baltico et al., « Practical functional encryption for quadratic functions with applications to predicate encryption », in : *Annual International Cryptology Conference*, Springer, 2017, p. 67-98.

[18] Raghavendran Balu et Teddy Furon, « Differentially private matrix factorization using sketching techniques », in : *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, 2016, p. 57-62.

[19] Elwyn R Berlekamp, « Algebraic coding theory mcgraw-hill », in : *New York* 8 (1968).

[20] Daniel Bernau et al., « Assessing differentially private deep learning with membership inference », in : *arXiv preprint arXiv :1912.11328* (2019).

[21] Daniel J Bernstein et Tanja Lange, « Computing small discrete logarithms faster », in : *Progress in Cryptology-INDOCRYPT 2012 : 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings 13*, Springer, 2012, p. 317-338.

[22] Dan Boneh et Xavier Boyen, « Secure identity based encryption without random oracles », in : *Annual International Cryptology Conference*, Springer, 2004, p. 443-459.

[23] Dan Boneh et Matthew Franklin, « Identity-based encryption from the Weil pairing », in : *SIAM journal on computing* 32.*3* (2003), p. 586-615.

[24] Dan Boneh, Amit Sahai et Brent Waters, « Functional encryption : Definitions and challenges », in : *Theory of Cryptography : 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, Springer, 2011, p. 253-273.

[25] Zvika Brakerski, Craig Gentry et Vinod Vaikuntanathan, « (Leveled) fully homomorphic encryption without bootstrapping », in : *ACM Transactions on Computation Theory (TOCT)* 6.*3* (2014), p. 1-36.

[26] Zvika Brakerski et Vinod Vaikuntanathan, « Circuit-ABE from LWE : unbounded attributes and semi-adaptive security », in : *Annual International Cryptology Conference*, Springer, 2016, p. 363-384.

[27] Zvika Brakerski et Vinod Vaikuntanathan, « Efficient fully homomorphic encryption from (standard) LWE », in : *SIAM Journal on computing* 43.*2* (2014), p. 831-871.

[28] Emmanuel Bresson, Dario Catalano et David Pointcheval, « A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications », in : *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2003, p. 37-54.

[29] Jan Camenisch et Victor Shoup, « Practical verifiable encryption and decryption of discrete logarithms », in : *Annual International Cryptology Conference*, Springer, 2003, p. 126-144.

[30] Hervé Chabanne et al., « Privacy-preserving classification on deep neural network », in : *Cryptology ePrint Archive* (2017).

[31] Kewei Cheng et al., « Secureboost : A lossless federated learning framework », in : *IEEE Intelligent Systems* 36.*6* (2021), p. 87-98.

[32] Martine de Cock et al., « Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data », in : *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, 2015, p. 3-14.

[33] Jean-Sébastien Coron, Tancrede Lepoint et Mehdi Tibouchi, « Practical multilinear maps over the integers », in : *Annual Cryptology Conference*, Springer, 2013, p. 476-493.

[34] Anamaria Costache et al., « Privacy-preserving distributed linear regression on high-dimensional data », in : *Cryptology ePrint Archive* (2022).

[35] Pratish Datta, Ratna Dutta et Sourav Mukhopadhyay, « Functional encryption for inner product with full function privacy », in : *Public-Key Cryptography–PKC 2016 : 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, Springer, 2016, p. 164-195.

[36] Whitfield Diffie et Martin Hellman, « New directions in cryptography », in : *IEEE Transactions on information theory* 22.*6* (1976), p. 644-654.

[37] Whitfield Diffie et Martin E Hellman, « New directions in cryptography », in : *Democratizing Cryptography : The Work of Whitfield Diffie and Martin Hellman*, 2022, p. 365-390.

[38] Edouard Dufour-Sans, Romain Gay et David Pointcheval, « Reading in the dark : Classifying encrypted digits with functional encryption », in : *Cryptology ePrint Archive* (2018).

[39] Cynthia Dwork, « Differential privacy : A survey of results », in : *International conference on theory and applications of models of computation*, Springer, 2008, p. 1-19.

[40] Cynthia Dwork, Aaron Roth et al., « The algorithmic foundations of differential privacy », in : *Foundations and Trends® in Theoretical Computer Science* 9.*3–4* (2014), p. 211-407.

[41] Cynthia Dwork, Guy N Rothblum et Salil Vadhan, « Boosting and differential privacy », in : *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, IEEE, 2010, p. 51-60.

[42] Mikhail Evchenko et al., « Frugal machine learning », in : *arXiv preprint arXiv :2111.03731* (2021).

[43] Sanjam Garg, Craig Gentry et Shai Halevi, « Candidate multilinear maps from ideal lattices », 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 32*, Springer, 2013, p. 1-17.

[44] Sanjam Garg et al., « Candidate indistinguishability obfuscation and functional encryption for all circuits », in : *SIAM Journal on Computing* 45.*3* (2016), p. 882-929.

[45] Sanjam Garg et al., « Functional encryption without obfuscation », in : *Theory of Cryptography : 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II 13*, Springer, 2016, p. 480-511.

[46] Adrià Gascón et al., « Privacy-preserving distributed linear regression on high-dimensional data », in : *Cryptology ePrint Archive* (2016).

[47] Romain Gay, Pierrick Méaux et Hoeteck Wee, « Predicate encryption for multi-dimensional range queries from lattices », in : *IACR International Workshop on Public Key Cryptography*, Springer, 2015, p. 752-776.

[48] Craig Gentry, « Fully homomorphic encryption using ideal lattices », in : *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, p. 169-178.

[49] Craig Gentry, Sergey Gorbunov et Shai Halevi, « Graph-induced multilinear maps from lattices », in : *Theory of Cryptography : 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II 12*, Springer, 2015, p. 498-527.

[50] Craig Gentry, Chris Peikert et Vinod Vaikuntanathan, « Trapdoors for hard lattices and new cryptographic constructions », in : *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, p. 197-206.

[51] Ran Gilad-Bachrach et al., « Cryptonets : Applying neural networks to encrypted data with high throughput and accuracy », in : *International conference on machine learning*, PMLR, 2016, p. 201-210.

[52] Shafi Goldwasser et al., « Multi-input functional encryption », in : *Advances in Cryptology–EUROCRYPT 2014 : 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, Springer, 2014, p. 578-602.

[53] Sergey Gorbunov, Vinod Vaikuntanathan et Hoeteck Wee, « Attribute-based encryption for circuits », in : *Journal of the ACM (JACM)* 62.*6* (2015), p. 1-33.

[54] Sergey Gorbunov, Vinod Vaikuntanathan et Hoeteck Wee, « Predicate encryption for circuits from LWE », in : *Annual Cryptology Conference*, Springer, 2015, p. 503-523.

[55] Vipul Goyal et al., « Attribute-based encryption for fine-grained access control of encrypted data », in : *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, p. 89-98.

[56] Vipul Goyal et al., « Bounded ciphertext policy attribute based encryption », in : *Automata, Languages and Programming : 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II 35*, Springer, 2008, p. 579-591.

[57] Rob Hall, Stephen E Fienberg et Yuval Nardi, « Secure multiple linear regression based on homomorphic encryption », in : *Journal of Official Statistics* 27.*4* (2011), p. 669-691.

[58] Stephen Hardy et al., « Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption », in : *arXiv preprint arXiv :1711.10677* (2017).

[59] Jeff Heaton, « Ian Goodfellow, Yoshua Bengio, and Aaron Courville : Deep learning : The MIT Press, 2016, 800 pp, ISBN : 0262035618 », in : *Genetic programming and evolvable machines* 19.*1-2* (2018), p. 305-307.

[60] *https ://librosa.org/doc/latest/index.html*.

[61] *https ://maritime.org/sound/.*

[62] *https ://matplotlib.org/.*

[63] *https ://scikit-learn.org/stable/.*

[64] *https ://www.sagemath.org/fr/.*

[65] *https ://www.tensorflow.org/ ?hl=fr.*

[66] Jinyuan Jia et al., « Memguard : Defending against black-box membership inference attacks via adversarial examples », in : *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, p. 259-274.

[67] Jonathan Katz, Amit Sahai et Brent Waters, « Predicate encryption supporting disjunctions, polynomial equations, and inner products », in : *Advances in Cryptology–EUROCRYPT 2008 : 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27*, Springer, 2008, p. 146-162.

[68] Jonathan Katz et Arkady Yerukhimovich, « On black-box constructions of predicate encryption from trapdoor permutations », in : *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2009, p. 197-213.

[69] Sungwook Kim, Jinsu Kim et Jae Hong Seo, « A new approach to practical function-private inner product encryption », in : *Theoretical Computer Science* 783 (2019), p. 22-40.

[70] Diederik P Kingma et Jimmy Ba, « Adam : A method for stochastic optimization », in : *arXiv preprint arXiv :1412.6980* (2014).

[71] Diederik P Kingma, Max Welling et al., « An introduction to variational autoencoders », in : *Foundations and Trends® in Machine Learning* 12.*4* (2019), p. 307-392.

[72] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton, « Imagenet classification with deep convolutional neural networks », in : *Advances in neural information processing systems* 25 (2012).

[73] Brenden Lake et al., « One shot learning of simple visual concepts », in : *Proceedings of the annual meeting of the cognitive science society*, t. 33, 33, 2011.

[74] Yann LeCun, Yoshua Bengio et Geoffrey Hinton, « Deep learning », in : *nature* 521.*7553* (2015), p. 436-444.

[75] Yann LeCun, Corinna Cortes, Chris Burges et al., *MNIST handwritten digit database*, 2010.

[76] Allison Lewko et al., « Fully secure functional encryption : Attribute-based encryption and (hierarchical) inner product encryption », in : *Advances in Cryptology–EUROCRYPT 2010 : 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, Springer, 2010, p. 62-91.

[77] Jiacheng Li, Ninghui Li et Bruno Ribeiro, « Membership inference attacks and defenses in classification models », in : *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, p. 5-16.

[78] Ninghui Li, Tiancheng Li et Suresh Venkatasubramanian, « t-closeness : Privacy beyond k-anonymity and l-diversity », in : *2007 IEEE 23rd international conference on data engineering*, IEEE, 2006, p. 106-115.

[79] Richard Lindner et Chris Peikert, « Better key sizes (and attacks) for LWE-based encryption », in : *Topics in Cryptology–CT-RSA 2011 : The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, Springer, 2011, p. 319-339.

[80] Qian Lou et al., « Glyph : Fast and accurately training deep neural networks on encrypted data », in : *Advances in neural information processing systems* 33 (2020), p. 9193-9202.

[81] Vadim Lyubashevsky, Chris Peikert et Oded Regev, « On ideal lattices and learning with errors over rings », in : *Advances in Cryptology–EUROCRYPT 2010 : 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, Springer, 2010, p. 1-23.

[82] Ashwin Machanavajjhala et al., « l-diversity : Privacy beyond k-anonymity », in : *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.*1* (2007), 3-es.

[83] Paulo Martins, Leonel Sousa et Artur Mariano, « A survey on fully homomorphic encryption : An engineering perspective », in : *ACM Computing Surveys (CSUR)* 50.*6* (2017), p. 1-33.

[84] Jonathan Masci et al., « Stacked convolutional auto-encoders for hierarchical feature extraction », in : *Artificial Neural Networks and Machine Learning–ICANN 2011 : 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, Springer, 2011, p. 52-59.

[85] Jose Maria Bermudo Mera et al., « Efficient lattice-based inner-product functional encryption », in : *IACR International Conference on Public-Key Cryptography*, Springer, 2022, p. 163-193.

[86] Daniele Micciancio et Oded Regev, « Worst-case to average-case reductions based on Gaussian measures », in : *SIAM J. Comput.* 37.*1* (2007), p. 267-302.

[87] Christoph Molnar, *Interpretable machine learning*, Lulu. com, 2020.

[88] Karthik Nandakumar et al., « Towards deep neural network training on encrypted data », in : *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019, p. 0–0.

[89] Milad Nasr, Reza Shokri et Amir Houmansadr, « Machine learning with membership privacy using adversarial regularization », in : *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, p. 634-646.

[90] Valeria Nikolaenko et al., « Privacy-preserving ridge regression on hundreds of millions of records », in : *2013 IEEE symposium on security and privacy*, IEEE, 2013, p. 334-348.

[91] Tatsuaki Okamoto et Katsuyuki Takashima, « Hierarchical predicate encryption for inner-products », in : *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2009, p. 214-231.

[92] Emilio Soria Olivas et al., *Handbook of research on machine learning applications and trends : Algorithms, methods, and techniques : Algorithms, methods, and techniques*, IGI global, 2009.

[93] Rafail Ostrovsky, Amit Sahai et Brent Waters, « Attribute-based encryption with non-monotonic access structures », in : *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, p. 195-203.

[94] Pascal Paillier, « Public-key cryptosystems based on composite degree residuosity classes », in : *International conference on the theory and applications of cryptographic techniques*, Springer, 1999, p. 223-238.

[95] Dingyi Pei, Arto Salomaa et Cunsheng Ding, *Chinese remainder theorem : applications in computing, coding, cryptography*, World Scientific, 1996.

[96] John M Pollard, « Kangaroos, monopoly and discrete logarithms », in : *Journal of cryptology* 13.*4* (2000), p. 437-447.

[97] Jianwei Qian et al., « De-anonymizing social networks and inferring private attributes using knowledge graphs », in : *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, IEEE, 2016, p. 1-9.

[98] Md Atiqur Rahman et al., « Membership Inference Attack against Differentially Private Deep Learning Model. », in : *Trans. Data Priv.* 11.*1* (2018), p. 61-79.

[99] Iyad Rahwan et al., « Machine behaviour », in : *Nature* 568.*7753* (2019), p. 477-486.

[100] Oded Regev, « On lattices, learning with errors, random linear codes, and cryptography », in : *Journal of the ACM (JACM)* 56.*6* (2009), p. 1-40.

[101] M Sadegh Riazi et al., « Chameleon : A hybrid secure computation framework for machine learning applications », in : *Proceedings of the 2018 on Asia conference on computer and communications security*, 2018, p. 707-721.

[102] Ronald L Rivest, Adi Shamir et Leonard Adleman, « A method for obtaining digital signatures and public-key cryptosystems », in : *Communications of the ACM* 21.*2* (1978), p. 120-126.

[103] Pierangelo Rosati et al., « Social media and stock price reaction to data breach announcements : Evidence from US listed companies », in : *Research in International Business and Finance* 47 (2019), p. 458-469.

[104] Bita Darvish Rouhani, M Sadegh Riazi et Farinaz Koushanfar, « Deepsecure : Scalable provably-secure deep learning », in : *Proceedings of the 55th annual design automation conference*, 2018, p. 1-6.

[105] Reza Shokri et al., « Membership inference attacks against machine learning models », in : *2017 IEEE symposium on security and privacy (SP)*, IEEE, 2017, p. 3-18.

[106] Peter W Shor, « Algorithms for quantum computation : discrete logarithms and factoring », in : *Proceedings 35th annual symposium on foundations of computer science*, Ieee, 1994, p. 124-134.

[107] Connor Shorten et Taghi M Khoshgoftaar, « A survey on image data augmentation for deep learning », in : *Journal of big data* 6.*1* (2019), p. 1-48.

[108] Latanya Sweeney, « k-anonymity : A model for protecting privacy », in : *International journal of uncertainty, fuzziness and knowledge-based systems* 10.*05* (2002), p. 557-570.

[109] Alaa Tharwat et al., « Linear discriminant analysis : A detailed tutorial », in : *AI communications* 30.*2* (2017), p. 169-190.

[110] Tijmen Tieleman, Geoffrey Hinton et al., « Lecture 6.5-rmsprop : Divide the gradient by a running average of its recent magnitude », in : *COURSERA : Neural networks for machine learning* 4.*2* (2012), p. 26-31.

[111] Erico Tjoa et Cuntai Guan, « A survey on explainable artificial intelligence (xai) : Toward medical xai », in : *IEEE transactions on neural networks and learning systems* 32.*11* (2020), p. 4793-4813.

[112] Junichi Tomida, Masayuki Abe et Tatsuaki Okamoto, « Efficient functional encryption for inner-product values with full-hiding security », in : *Information Security : 19th International Conference, ISC 2016, Honolulu, HI, USA, September 3-6, 2016. Proceedings 19*, Springer, 2016, p. 408-425.

[113] Stacey Truex et al., « A hybrid approach to privacy-preserving federated learning », in : *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019, p. 1-11.

[114] Marten Van Dijk et al., « Fully homomorphic encryption over the integers », in : *Advances in Cryptology–EUROCRYPT 2010 : 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, Springer, 2010, p. 24-43.

[115] David A Van Dyk et Xiao-Li Meng, « The art of data augmentation », in : *Journal of Computational and Graphical Statistics* 10.*1* (2001), p. 1-50.

[116] Ashish Vaswani et al., « Attention is all you need », in : *Advances in neural information processing systems* 30 (2017).

[117] Naga Vemprala et Glenn Dietrich, « A social network analysis (SNA) study on data breach concerns over social media », in : (2019).

[118] Yaqing Wang et al., « Generalizing from a few examples : A survey on few-shot learning », in : *ACM computing surveys (csur)* 53.*3* (2020), p. 1-34.

[119] Brent Waters, « Ciphertext-policy attribute-based encryption : An expressive, efficient, and provably secure realization », in : *International workshop on public key cryptography*, Springer, 2011, p. 53-70.

[120] Brent Waters, « Dual system encryption : Realizing fully secure IBE and HIBE under simple assumptions », in : *Annual International Cryptology Conference*, Springer, 2009, p. 619-636.

[121] Hoeteck Wee, « Functional encryption for quadratic functions from k-lin, revisited », in : *Theory of Cryptography : 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I 18*, Springer, 2020, p. 210-228.

[122] Gilbert Wondracek et al., « A practical attack to de-anonymize social network users », in : *2010 ieee symposium on security and privacy*, IEEE, 2010, p. 223-238.

[123] Runhua Xu, James BD Joshi et Chao Li, « Cryptonn : Training neural networks over encrypted data », in : *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2019, p. 1199-1209.

[124] Runhua Xu et al., « Hybridalpha : An efficient approach for privacy-preserving federated learning », in : *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019, p. 13-23.

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

NAVAL GROUP

Titre : Méthodologie de conception de modèles d'apprentissage automatique sur données à caractère confidentiel

Mot clés : Chiffrement Fonctionnel, PPML, Apprentissage automatique, Learning With Error

Résumé : Dans cette thèse nous étudions un ensemble de techniques relatives à l'application du chiffrement fonctionnel pour la conception de modèles d'apprentissage automatique sur données à caractère confidentiel. Après avoir passé en revu les schémas de chiffrement fonctionnels majeurs de l'état de l'art, nous proposons des combinaisons et des adaptations de schémas existants. Nous portons une attention particulière aux schémas basés sur les problèmes de réseaux LWE et RLWE pour leurs propriétés post-quantiques. A des fins d'expérimentations, nous déroulons des scénarios d'utilisation du chiffrement fonctionnel pour la conception et l'exploitation de modèles de classification sur des données de type images et acoustiques. Nous démontrons ainsi la faisabilité d'application de ce type de construction cryptographique à des cas d'usage industriels. En particulier, nous rapportons des exemples de déchiffrements fonctionnels permettant le calcul d'analyse par composantes principales, d'analyse discriminante linéaire et de transformée de Fourier, ainsi que le calcul de couches neuronales de convolutions linéaires et quadratiques. Enfin, nous proposons un ensemble de méthodes pour la recherche de paramètres de chiffrement fonctionnel maximisant les performances des modèles à apprendre tout en garantissant un niveau de sécurité et une probabilité d'exactitude de déchiffrement.

Title: A methodology for machine learning models training on confidential data

Keywords: Functional Encryption, PPML, Machine Learning, Learning With Error

Abstract: In this thesis, we study a set of techniques for functional encryption application for machine learning models training on confidential data. After reviewing the major state-of-the-art functional encryption schemes, we propose combinations and adaptations of existing schemes. We pay particular attention to those based on lattice-based problems LWE and RLWE for their post-quantum properties. For experimental purposes, we unfold scenarios using functional encryption to train and use classification models on image and acoustic data. We demonstrate the feasibility of applying this type of cryptographic construction to industrial use cases. In particular, we report examples of functional decryptions for the computation of principal component analysis, linear discriminant analysis and Fourier transform, as well as the computation of neural layers of linear and quadratic convolutions. Finally, we propose a set of methods for finding functional encryption parameters that maximize the performance of models to be learned, while guaranteeing a given level of security and probability of decryption correctness.