# Contributions to data confidentiality in machine learning by means of homomorphic encryption

Martin Zuber

université
PARIS-SACLAY

# Contributions to data confidentiality in machine learning by means of homomorphic encryption

**Thèse de doctorat de l'université Paris-Saclay**

École doctorale n° 580, sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Mathématique et Informatique
Unité de recherche: Université Paris-Saclay, CEA, Département Systèmes et Circuits Intégrés Numériques (DSCIN), 91191, Gif-sur-Yvette, France
Référent: : Faculté des sciences d'Orsay

**Thèse présentée et soutenue en visioconférence, le 10 décembre 2020, par**

**Martin ZUBER**

**Composition du jury:**

| | |
|---|---|
| **Caroline Fontaine**<br>Directrice de Recherche CNRS, École Normale Supérieure de Paris-Saclay | Présidente |
| **Philippe Gaborit**<br>Professeur des Universités, Université de Limoges | Rapporteur et Examinateur |
| **David Pointcheval**<br>Directeur de Recherche CNRS, Ecole Normale Supérieur de Paris | Rapporteur et Examinateur |
| **Mariya Georgieva**<br>Experte Cryptographie, INPHER | Examinatrice |
| **Pascal Paillier**<br>Expert Cryptographie, CryptoExpert | Examinateur |
| **Renaud Sirdey**<br>HDR, Directeur de Recherche CEA, CEA Paris-Saclay | Directeur |

Thèse de doctorat

NNT: 2020UPASG048

# Résumé

L'objectif de mes travaux tout au long de cette thèse a été de permettre à des algorithmes complexes d'apprentissage machine de pouvoir être appliqués (lors de leur phase d'inférence) sur des données dont la confidentialité est préservée. Un contexte d'application est l'envoi de données sur un serveur distant sur lequel un algorithme est évalué. Selon les cas, pour des raisons éthiques, légales ou commerciales, la confidentialité des données qui sont envoyées doit pouvoir être respectée. Il est possible pour cela de désigner des autorités en lesquels tous les acteurs du protocole peuvent avoir confiance. Pourquoi accorder à des entités un tel niveau de confiance dans des cas où la confidentialité des données d'un utilisateur est essentielle ? La cryptographie offre en effet des alternatives, dont le chiffrement totalement homomorphe.

Le chiffrement homomorphe permet, en théorie, l'évaluation de n'importe quelle fonction dans le domaine chiffré. Son utilisation peut donc être imaginée dans le cas ou un utilisateur envoie des données chiffrées sur un serveur distant qui détient un algorithme puissant d'apprentissage machine. La phase d'inférence de cet algorithme est alors effectuée sur donnes chiffrées et le résultat est renvoyé à l'utilisateur pour déchiffrement. La cryptographie propose d'autre méthodes de calcul sur données chiffrées qui sont présentées succintement dans le manuscrit. Pour faire court, la particularité du chiffrement homomorphe est qu'il ne nécessite aucune intéraction entre l'utilisateur et le serveur.

Dans ma thèse, je présente trois principaux algorithmes d'apprentissage machine sécurisés: une évaluation sur données et modèle chiffrés d'un réseau de neurone récursif et discret, le réseau de Hopfield; une reconnaissance de locuteur sur modèle chiffré pour un réseau auto-encodeur, le système VGGVox; une évaluation sur données chiffrées d'un classifieur des k plus proches voisins (ou classifieur k-NN). Notamment, notre classifieur k-NN sécurisé est le premier tel algorithme évalué de manière totalement homomorphe. Nos travaux ouvrent de nombreuses perspectives, notamment dans le domaine de l'apprentissage collaboratif sécurisé.

# Acknowledgements

First and foremost, let me thank Renaud Sirdey, who supervised this thesis. I don't know how we managed it, but it looks like this Ph.D. went smoothly. A little too much maybe: how can I tell my future children that I had to fight tooth and nail through three years of hell to get it ? I guess I can lie.. Anyway, yay for us ! Seriously though, I mainly have you to thank for that, your trust and your advice. Looking forward to working together some more !

I would like to extend my sincere thanks to David Pointcheval and Philippe Gaborit for reviewing this thesis. I would like to think that it wasn't too much of a burden (it being so well written and all) yet I realize the amount of work that gets into it and the importance of that work for the research community as a whole. For this and their kind words at the Ph.D. defense, thank you.

I have Caroline Fontaine, Ph.D. committee president, to thank for a smooth and orderly Ph.D. defense in these trying times. I extend the same gratitude to Mariya Georgieva and Pascal Paillier, members of my Ph.D. committee. I was flattered by the interest that you showed in my work and hope that we can have more such occasions to discuss on similar matters.

To everyone at CEA, thank you for making the place a welcoming work environment. I still think the coffee is not that good, but the selection of teas and infusions is quite appropriate. On the matter of coffees, I'd like to extend a special thank to my coffee-loving, cigarette-smoking, PhD-writing, office-sharing friend Daniel Vert. To all the other Ph.D. students that have crossed my path at CEA I would like to say thanks for the motivation and the interesting conversations. Hoping I'm not forgetting anyone: Briag Le Nabec, Émine Laarouchi, Farouk Hebbache, Phlippe Glanon, Abbass Madi, Gabriella Bettonte, Benjamin Binder. To Sergiu Carpov, here's hoping we can work together again some day, you've got a great brain worth picking. To the guys at the DM2I department - Arnaud Grivet Sébert, Rafael Pinot and Cédric Gouy-Pailler - thanks for the opportunity to work with you. It was great to know that my work could be useful to others and in the service of an interesting and novel idea.

To Mélanie, thank you for your support and patience. Now that this is done, we can focus on *le Gros* with greater resolve. We can expect many more challenges ahead and I look forward to our common struggles.

To mamma, I am finally a doctor though not exactly of the type you would have hoped for - though I have the stethoscope. You taught me both the value of self-care and that it's okay to fail, which made it all the easier to succeed. Dad, remember when I said I would never be doing research ? That has not aged well. This is in great part thanks to you. I have looked up to your spirit of research and your determination all my life. Thomas, I beat you to it. You have been the person I have been trying to emulate since I was born, copying your every move from taking

piano lessons to climbing. For many years you have been the definition of what I could hope to achieve if I put my mind to it. It must not have been easy having someone look up to you that much and for that I'm sorry ! Gros Bébé, I didn't beat you to it. You have had your Ph.D. in astrophysics for a while and now you have turned your attention to other deserving subjects like rap and kebabs. Jokes aside, I have the greatest faith in your finding your way in life far from the roads that your brothers walked before you. It must not and it will not be easy to be in your position, but you have an extraordinary strength and resolve.

A special mention is due to Pierre and David, great flatmates and friends. You have been a source of great support through this.

Finally, I have Dominique Thonier and Arnaud Basson to thank for being the best teachers I could ever have hoped for, each of them at the right time in my life.

# Contents

# Chapter 1

# Introduction

## 1.1 The grand scheme of things

**DNA-matching in the U.S. and China** On June 29$^{th}$ 2020, Joseph James DeAngelo, 74, pleaded guilty to 13 murders and nearly 50 rapes committed across California in the 1970s and '80s. Known before 2018 only as *The Golden State Killer*, his arrest that year was only made possible by a novel DNA technique matching DNA found on crime scenes on genealogy websites in order to link violent crimes to relatives of the culprit. His arrest and subsequent guilty pleas were a sigh of relief for the families of his victims longing for justice.
Following the success of its use in the Golden State Killer case, this novel DNA database matching technique is now being used widely in the United States. It has lead to a series of breakthroughs in cases previously thought unsolvable. In this context, objecting to its use can seem irresponsible as its social utility - catching violent criminals - has been affirmed without a doubt. Yet concerns remain.

Ethical questions on the matter are aplenty. First of all, about the creation of the DNA databases themselves: does one own their DNA and can they sell it or give it away without consent from their immediate family ? Giving away a sample of my blood for DNA collection may have repercussions for all of those who share important parts of my DNA. Governments around the world can start requiring DNA samples from their citizens. On the use of the existing DNA databases: who can have access to DNA databases and for which purposes ? There are myriad organizations and institutions which would benefit from an access to DNA databases of a population, from different branches of the government to private companies such as insurance providers.

The use of these techniques is increasing across the world, making these ethical questions all the more pressing. Nowhere are they more pressing than in China. In Proust's *Albertine Disparue*, the socialite M$^{me}$ de Guermantes offers an enigmatic and absurd answer to a tame soul inquiring about the nature of her worry: *"La Chine m'inquiète"*[1]. A prescient answer if there ever was one in light of recent developments. According to a report by the International Cyber Policy Center [DL20], since late 2017, the police in China have been collecting enough blood samples from men and boys across the country to build a vast DNA database. They estimate the number of DNA profiles collected thus far to between 100 and 140 million, making it the world's largest DNA

[1] China worries me

database. The use of a specific DNA matching technique called Y-STR allows close relatives to be matched even when their DNA was not collected. Consequently, the report points out that the province of Henan was able to achieve close to total genetic coverage by retrieving samples from 10% of the male population.

On the "old continent", ethical matters surrounding the use and storage of data are being addressed with more zeal - it seems - than anywhere else. Recently, the General Data Protection Regulation (GDPR) has brought these issues to light in an unprecedented way and with immediate international consequences. Well-meaning governments and international organizations are not sufficient to prevent misuse and abuse however. This is true even in the case when legislation is crafted with care and insight and with sufficient means allocated to enforced it. Indeed, no legislation can realistically be expected to cover the specific problems that will arise from future research developments (medical recommendations from genetic analysis for instance in our case). For this reason, a constant public debate on the subject is essential. So are a large set of confidentiality-preserving tools from which to pick and choose to fit any specific ethical problem that arises.

The use of DNA matching techniques can be useful in at least one universally recognizable setting (catching violent criminals). It can also be socially destructive for societies aiming to respect the freedom of their citizens. In any case, it's use is growing, and the many ethical matters surrounding it need to be addressed. Those ethical matters - and DNA matching techniques - are not precisely the focus of this manuscript. They serve to illustrate the need for innovations in the field of cryptography, and present challenges that our own work partially addresses.

**The purpose of this thesis.** We aim to provide a set of frameworks allowing for complex functionalities (DNA matching is an example of those) to yield their intended results while ensuring confidentiality properties are achieved. For instance, if a society were to adhere to the use of DNA matching techniques for the solving of violent crimes, it could reasonably expect the DNA database collected for that purpose be used *only* for that purpose. This can be achieved through regulatory measures such as prohibiting the use of the database in certain cases and restricting its access to certain law enforcement agencies. However, this means that the DNA database and the DNA samples tested for matches will - at some point - be used by a single entity which must be trusted absolutely to abide by the current regulations.

The fundamental reason for the existence of our work - and every other work like it - is the following: why trust that an outside entity will not misuse personal data when you can have assurances of that fact ? This applies both in the case of a private company that may use/sell your data for profit, legally or illegally. It also applies to use by a government which may or may not have the proper safeguards against abuse, as well as the proper security for data storage and access.

There are a multitude of ways to ensure *security* in the case of an *outsourced computation*. One reason for this is that there are more than one problem to solve. What are we protecting exactly ? Is it personal user data or complex algorithms developed by a company ? Do we need/want to hide all aspects of the data or merely mask its origins ? Who is trying to steal what and with which means ? The questions could go on and on and provide us with two very clear objectives for this manuscript:

- We will strive to explain to the best of our ability what the various stakes involved in *secure outsourced computations* are (as well as define precisely what we mean by this vague term).

- We will present the technical contributions that we have made to this field and strive to place them in the general context of the existing literature.

## 1.2 The specific scheme of things

Precisely, most of our work focuses on finding new algorithms for *secure outsourced machine learning evaluation* using *Fully Homomorphic Encryption (FHE)*. All of these terms will be defined and commented on in more detail further in this thesis. However, for context, here is a short description of every one of them, as they can be defined in the literature, and as we chose to interpret them in our work:

**Machine learning.** We restrict our research to applications in the realm of *machine learning*. However, given the size of that field and the number of algorithms that fall into that category, we are not so much "restricting our research" as we are using it to explore small parts of that field in the search of secure applications. To be precise, we will mainly present three secure machine learning algorithms developed in the course of our research. All of these algorithms are used to classify data points: attach a pre-determined class to each point. Classification-oriented machine learning is not the only existing kind of machine learning, but it is the most common, and therefore the one we focus on throughout our work.

**Evaluation.** Machine learning algorithms usually have two distinct phases: the training phase and the *evaluation* phase. The reality is sometimes more subtle, for instance an algorithm can consistently be trained using results from the evaluation phase. The evaluation phase consists of applying a classification algorithm over some data points. During the training phase, the model (everything that defines the classification algorithm) is built through a learning algorithm. There are many aspects of the training phase that are worth protecting from prying eyes: the training algorithm, the topology of the underlying structure (in the case of neural networks, the number of layers and the number of neurons per layers), the data used for the training. Ensuring security during the training phase is usually regarded as a tougher challenge than during the evaluation phase. Whether this changes with future research is anyone's guess. However it is clear that the challenges secure training poses are much different than the ones pertaining to a secure evaluation. In our work, we focus mostly on the evaluation phase, and only touch briefly on some aspects of securely training a machine learning algorithm.

**Outsourced computation.** An *outsourced computation* is one that needs to use data and/or other sensitive information (such as the nature of the computation) from two distrusting sources at two locations. The computation can then happen following such protocols as: computing in tandem between the two parties; computing exclusively on one of the locations; giving a third party the responsibility for the computation. Any one of those protocols (and more) can be imagined and there are different tools to implement them. How interested parties decide to move forward will depend on the properties they wish to achieve (which kind of security, on what) given their constraints (power consumption, computational limits, bandwidth usage, network latencies,..). In this manuscript, we focus on the case of a computation being made at one of the locations with no interaction between the parties during the computation. We will explain this choice extensively, and mention briefly the solutions that exist to implement other protocols.

**Secure.** *"Secure"* is an abstract concept, reassuring and yet vague. In the realm of cryptography in particular, evaluating what is secure and to what degree is paramount. Any cryptographic tool is only secure for a given threat model with a computationally-bounded adversary. In the realm of outsourced computations, "security" can be deployed over a number of items, and in a number of ways. As mentioned before, there are different aspects of an outsourced computation one could wish to hide: any entity's data being sent on a distant server; the nature of the computation; the parameters used for the computation. On top of that, this "security" can take several forms. Essentially, in this thesis, we will make a clear distinction between the concepts of data-confidentiality (hiding all aspects of a dataset) and data-privacy (hiding the origins of a dataset). We focused our work on achieving data-confidentiality (a stronger property) and therefore limit most of this manuscript (with some exceptions) to such cases.

**Fully homomorphic encryption.** There are several ways to ensure data-confidentiality over an outsourced computation. We will present the main options available in the hope that our choice to use *FHE* as a direction for our research is made clear. In short, FHE is a relatively novel cryptographic tool that saw its first theoretical breakthrough a little over ten years ago. The idea is simple: take an existing algorithm over clear data, encrypt the data, run the algorithm over the encrypted data, decrypt the output, et voilà ! The decrypted output should be the same as the one we would obtain by running that algorithm on clear, non-encrypted inputs. In practice, unfortunately, FHE has not yet come to a point where it can be applied as thoughtlessly. As it is an integral part of our research, FHE in general and one encryption scheme in particular - TFHE - will be presented in much greater detail in the rest of the thesis.

## 1.3 Cryptographic schemes

Cryptography has historically been used to communicate securely over insecure channels. At the start, Alice (A) - as per tradition - has a *clear* (or *plaintext*) message which is *encrypted* into a *ciphertext*. That process - the encryption process - is originally one that involves a *secret key*, known only to Alice, and is such that no one should be able to find the original plaintext given *anything less* than the ciphertext and the secret key. By sharing that secret key with Bob (B) on a secure channel (in person for instance), Alice can then send ciphertexts to Bob in plain view of everyone knowing that only Bob can apply the *decryption* algorithm accurately and retrieve the plaintext Alice intended to share with him.

A number of such encryption schemes have been used over the centuries, evolving to match the constraints of their time. For a long time, they all matched the same formula: they were *private-key encryption schemes* for the very understandable reason that the secret key had to remain secret for the scheme to work. With the inception of *public-key encryption schemes* in 1976 [DH06], cryptography opened its doors to many more applications, not just simple communication tasks. Beyond message transfers, cryptography is now known for digital signature schemes [RSA78], key management, data integrity techniques, message authentication to name a few applications.

At the turn of the 21$^{st}$ century, the field of cryptography has seen a host of additions around the topic of secure computations. FHE is only one of a number of such proposed solutions to a problem that was once seen as an out-of-reach challenge. These novel cryptographic tools include - among others - Functionnal Encryption (FE), Identity-Based Encryption (IBE), Attribute-Based Encryption (ABE) and Multi-Party Computation (MPC). Theses tools have been growing in popularity and recent developments have pushed them closer than ever to widespread real-

world use.

## 1.4 Our contributions

The work that we present here can be summarized in a few words. We propose novel, non-interactive, fully homomorphic alternatives to three existing machine learning algorithms: Hopfield networks, embedding-based neural networks, $k$-NN classifiers. To do this, we build on existing FHE tools that we tweak to our convenience.

**An homomorphic Hopfield network** The first work that we present is described in Chapter 6. It was published as [ISZ19] and presented at the CANS 2019 conference. This first work was inspired by a desire to test relatively unknown machine learning algorithms for homomorphic applications. We eventually settled on the use of a Hopfield network because of its FHE-friendly structure. We trained and tested our own network on several databases and chose face-recognition as good application to showcase our work. Our Hopfield network uses a set of correlation values (its weights) to store a number of "model faces" with which a new face will be compared. That new face will be classified as belonging to the same person as one of the model faces. In fact, faces are not stored "as is". There is a pre-processing step used on any picture to extract a vector that accurately describes the features of the face in that picture. That is the vector used for storage by the network and then recognition. Our work was novel in that it provided a scheme where both the weights of the network and the input vector to be classified could be encrypted (at the same time).

**An homomorphic argmin operator** When assessing the strengths of Hopfield networks with regard to homomorphic applications, one aspect that we found of interest was the relatively easy training algorithm that is required. That algorithm simply consists of a set of scalar product operations and, as any training algorithm, a selection phase. That selection phase requires the use of an argmin operator (to select the model with the smallest error rate) or - alternatively - an argmax operator (to select the model with the highest classification rate).

Because of this, our interest was directed toward the production of a fully-homomorphic non-interactive argmin operator. The results stemming from this work were published as [ZCS20] and presented at the ICICS 2020 conference. They are described in Chapter 7. Our goal when thinking about our work's presentation was to showcase the performance of our argmax operator while at the same time pursuing our original research endeavor to search for FHE-friendly machine learning algorithms. We did this by using our homomorphic argmin operator to achieve model-confidentiality over an embedding-based neural network speaker-recognition system: VGGVox. This system uses complex machine learning algorithms (Convolutional Neural Networks or CNNs) to transform raw audio data into vectors in a space where the Euclidean distance is a good measure of speaker similarity. When two vectors are close in the output Euclidean space, one can infer that their corresponding raw audio data originated from the same speaker. To classify a new audio as coming from a specific speaker stored by the system, one needs to determine which of the stored model vectors are closest to the new vector. It is in this comparison phase that our argmin operator is used.

**An homomorphic k-NN classifier** An argmin operator can be used very efficiently in one of the simplest machine learning algorithms: the 1-Nearest Neighbor (1-NN) classifier. Essentially, this algorithm works exactly like the VGGVox system but does not use its complex transformation phase with CNNs. An application of the 1-NN classifier in the same context would have one

compute distances between the raw audio data directly an then conclude based on the smallest distance. This is obviously impossible when the input vectors do not have the same sizes. While this is not the only issue 1-NN classifiers face, depending on the nature of the data, they remain among the simplest ways to analyze data, and a surprisingly efficient way as well. This is why a slightly more complicated version - the $k$-NN classifier - is still widely used in practical applications today, with the notable mention of recommendation systems (purchase or viewing suggestions based on the user's previous behavior). The $k$-NN classifier applies the same principle as the 1-NN classifier but produces a result based on a majority vote between the $k$ closest neighbors to our new vector.

In Chapter 8, we present our work adapting the argmin operator from Chapter 7 to produce a fully homomorphic non-interactive $k$-NN classifier. This work was accepted at Proceedings on Privacy Enhancing Technologies 2021.

## 1.5 Manuscript structure

This thesis is composed of two parts. One that presents a general and technical overview of the landscape in which our work was done, and another which presents our contributions to that landscape.

Part I consists of a presentation of the context surrounding our work. Chapter 3 presents the general notions that are touched on in our work: machine learning and confidentiality-preserving methods. It aims to present a broad overview of the field and not just the scope of our contributions. This will help the reader understand our choices and the constraints that we faced. In Chapter 4, we present some of the existing work relating to our contribution. We both cite the most relevant works on the subject of homomorphic encryption for machine learning and strive to paint an accurate general picture of the existing literature on the larger matter of confidentiality-preserving machine learning. Chapter 5 is dedicated to presenting the technical notations that we use throughout the thesis, as well as essential technical background on the FHE schemes that we use.

Part II consists of three chapters, each of them presenting one of our contributions. Each chapter corresponds to an article that was either published (Chapters 6 and 7) or was accepted for publication at a journal at the time of writing (Chapter 8). Chapter 6 presents our homomorphic Hopfield network. Chapter 7 presents our homomorphic argmin operator and its application in a model-confidential embedding-based deep neural network speaker recognition setting. Chapter 8 presents our homomorphic $k$-NN classifier.

# Chapter 2

# Personal Publications

- Malika Izabachène, Renaud Sirdey, and Martin Zuber. "Practical Fully Homomorphic Encryption for Fully Masked Neural Networks". In: Cryptology and Network Security. Ed. by Yi Mu, Robert H. Deng, and Xinyi Huang. Cham: Springer International Publishing, 2019, pp. 24–36. isbn: 978-3-030-31578-8.
- Martin Zuber, Sergiu Carpov, and Renaud Sirdey. "Towards Real-Time Hidden Speaker Recognition by Means of Fully Homomorphic Encryption". In: Information and Communications Security. Ed. by Weizhi Meng, Dieter Gollmann, Christian D. Jensen, and Jianying Zhou. Cham: Springer International Publishing, 2020, pp. 403–421.
- Martin Zuber and Renaud Sirdey. "Efficient homomorphic evaluation of k-NN classifiers". In: Proceedings on Privacy Enhancing Technologies (2021).
- Arnaud Grivet Sébert et al. "SPEED: Secure, PrivatE, and Efficient Deep learning". In: CoRR abs/2006.09475 (2020). (submitted for publication).

# Part I

# Context and Motivations

# Chapter 3

# General notions

**Contents**



In this chapter, we will introduce the general notions surrounding confidentiality, machine learning, cryptography. Some of these concepts will require more in-depth study, specifically with regard to the most recent work done on the subject. When that is the case, the notion will be studied further in Chapter 5.

## 3.1 Confidentiality and Privacy

In the literature, confidentiality and privacy can be found to be defined in a number of ways. These terms have specific legal and/or ethical definitions in different countries. In the field of information security, they are generally thought to define the same property: "keeping information secret from all but those who are authorized to see it" according to [MVO96]. In the wider context of information security, it is but one of many such properties one could wish to obtain in a given setting (data integrity, signature, authentication,..).

In the context of outsourced computation however, in this thesis, we believe that a more precise distinction needs to be made. We will speak of confidentiality with respect to data that needs to be hidden, and privacy with respect to an entity that wishes to not be linked to its personal data. Thusly defined, confidentiality is a stronger property than privacy: if one's personal data is completely obfuscated, it cannot be linked to its source; one could however erase the link

between clear data and its source and thus obtain privacy without confidentiality. That last property can only be obtained if the data is not "naturally" linked to its origin as is the case with DNA.

The general context in which we are looking at privacy and confidentiality here is that of an *outsourced computation*: for legal, financial or practical reasons an entity can wish to outsource a computation over sensitive data. In this context, there are several confidentiality and privacy properties one could want to obtain. To give the reader some context, we now present a possible example of outsourced computation.

Take the case of a database containing confidential information: the *data*. It was obtained by collecting personal *records* (e.g. multi-dimensional vectors) from a set of *users* and is stored on a secure *server*. Each record is made up of several *attributes* (e.g. the elements of the vectors). An outside entity - the *client* - wishes to analyze the data with a complex *computation* (an algorithm) that requires an input: the *query* (e.g. a multi-dimensional vector made up of attributes). There are many ways to allow the client to obtain its desired result depending on the confidentiality constraints that we have. Some methods are more complex than others, each have their drawbacks and advantages. In this simple context, there are a number of confidentiality properties one could want to achieve, for a number of different reasons. Below are a set of such properties, numbered from 1 to 7.

- *record confidentiality.* Whereby no one but the server can have access to the records in clear form. One reason for record confidentiality is the usually high cost of acquiring a good database. Another reason is legal constraints: organizations may not be at liberty to give away or sell private information without a user's consent, even if that information is not linked to the user from which it was obtained. (1)
- *user privacy.* Whereby no one but the server can link a given user to its personal data in clear form. In other words, no one can have access to both the data and the identity of the corresponding users at the same time. It is a strictly weaker constraint than record confidentiality as one could achieve user privacy but not record confidentiality. The reverse is not true. (2)
- *attribute privacy.* Similar to user privacy, however we consider a harder constraint whereby no one can link a single user to the value of any attribute. (3)
- *query confidentiality.* Similar to record confidentiality, but with respect to the client. Whereby the nature of the query is known only to the client. This can be important in the case where the computation is a comparison of the client's personal data (the query) with the server's data. (4)
- *client privacy.* Similar to user privacy, but with respect to the client. Whereby no one can have access to both the contents of the query and the identity of the client. (5)
- *computation confidentiality.* Whereby only the server knows the nature of the computation it applies to the data. This can be important in the case where the nature of the computation reveals information about an algorithm that is expensive to obtain. (6)
- *result confidentiality.* Whereby only the client knows the final result. In the case where the computation is a remote and automated diagnosis over the client's personal medical data (the query), the result then amounts to a diagnosis, a very sensitive information. (7)

Table 3.1 presents a set of different "naive" approaches to the problem at hand. Each of them - at first glance - achieves different confidentiality and privacy properties, but not all. Ideally, we

| Setup | (1) | (2) | (3) | (4) | (5) | (6) | (7) |
|---|---|---|---|---|---|---|---|
| The client sends its query in clear form. The server applies the computation and sends the result to the client. | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ |
| The server sends the data to the client in clear form. The client applies the computation itself. | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ |
| The server sends the (perfectly anonymized) data to the client who applies the computation itself. | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |
| The ideal scenario. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 3.1: This table presents different naive protocols and their impact on the privacy and/or confidentiality of the data involved. Anonymization defines a process by which privacy is obtained for a set of data. See Section 3.3.3 for more detail.

would be interested in solutions that achieve all of the confidentiality properties above. There are limits to what cryptography alone can do regarding these properties, we will touch on that matter as well. This table is presented as a naive first approach to the problem. Table 3.2 is an updated, more accurate version of this one.

## 3.2 Machine Learning

Machine Learning (ML) is the study and design of algorithms that allow a computer to solve a problem without being given specific instructions on how to solve it. It is a wide area of research that has existed for over 60 years and only recently evolved to include the best known Deep Convolutional Neural Networks.

Machine learning algorithms can take many shapes. One of the most significant distinction is whether the "learning" is *supervised* or *unsupervised*, meaning whether the data used to train the algorithm is labeled or not. A machine learning algorithm consists of a *model* to be trained over a dataset before that model can be used for classification. The accuracy of a given model depends on the classification algorithm, the training algorithm, and the dataset used for training. In the case of a supervised machine learning algorithm, the two main phases of its use can be described as follows.

- *Training phase.* When given dataset to train a machine learning algorithm on, the standard approach is to first separate that dataset into a training set and a testing set. The idea is that the model be trained *only* using the training set and never using the testing set. Once the model has been trained and the trainer thinks it has achieved the best possible result, the model is then tested once on the testing set. Its performance on the testing set is what determines its accuracy. Figure 3.1 represents the training phases of a standard machine learning algorithm.
- *Inference phase.* Once the model has been trained, it can be used for classification over any single data point or set of data points for real-world applications. The inference phase can be repeated as long as there are users interested in using the trained model.

In some cases, when the structure of the machine learning algorithm permits it, results from the inference phase (user feedback) can be fed back to the model for a constant learning process. In any case, training a good, quality model has become increasingly reliant on acquiring a vast amount of data. Any machine learning algorithm has an inherent limit to what it can learn. One cannot simply keep throwing training data at it and expect the overall classification rate to keep rising. In the early days of machine learning, the structural limits on the learning of algorithms was such that not a great amount of data was needed to train them. Recently, and mostly with the inception of deep learning algorithms and convolutional neural networks, the amount of data required to properly train a state-of-the-art machine learning algorithm has increased dramatically. To a point that access to vast amounts of data has become a very valuable asset for a company or a government.

The focus of this work is not machine learning per se, but rather the means to obtain secure machine learning applications. We achieve our goal by using several different machine learning algorithms: Hopfield networks, embedding-based neural networks and $k$-NN classifiers. These algorithms are presented in detail in their respective chapters. What they have in common is their FHE-friendly nature. The use of FHE is both limited and can incur a high computational overhead. That means it cannot be used on any machine learning algorithm indiscriminately.

**About feed-forward neural networks.** There are a host of other machine learning algorithms, and one example worth mentioning is that of *feed-forward neural networks*. Even though our work does not concern them, they are the focus of a majority of the works similar to our. The reason for this is their widespread use and recognition around the world, both in research contexts, and in practical applications. They are a subset of the larger group of Artificial Neural Networks (ANNs) and come in a wide variety of flavors (fully-connected, convolutional, deep, ...). To be succinct, these neural networks are trained with an algorithm much too complex for any modern FHE scheme to be able to adapt: the *back-propagation* algorithm or *gradient descent*. The evaluation phase consists of a series of scalar product operations followed by an application of the *activation function*, a complex non-linear function such as a sigmoid ($\theta(x) = (1 + e^{-x})^{-1}$) or a rectified linear function ($\theta(x) = \max(0, x)$) also known as ReLU. We only mention this to give the reader some context on the literature we cite in Chapter 4.

## 3.3 Confidentiality for Machine Learning

### 3.3.1 The need for confidentiality

As we mentioned in our introduction, the use of DNA-matching techniques can provide a great service to society as a whole. Because their use can also present risks for societies that value individual freedom, a choice can be made to prevent data collection in the name of privacy. Because we are idealists, we believe we can have the best of both worlds: ensure privacy and/or confidentiality over sensitive data used during the training or evaluation phase of a machine learning algorithm for a good cause.

There are cases, however where achieving confidentiality properties over the training or the use of a machine learning algorithm does not seem to matter at first. This can happen when the training and inference data is highly non-sensitive: in the case of pictures of leaves for a plant-recognition software for instance. However, building such a software requires machine learning know-how and time. For this reason, the owner has an interest in protecting the model that she trained from being stolen. If the owner of the model is also the entity performing the inference on their own server, then protecting the model means ensuring the server is secure. In the case

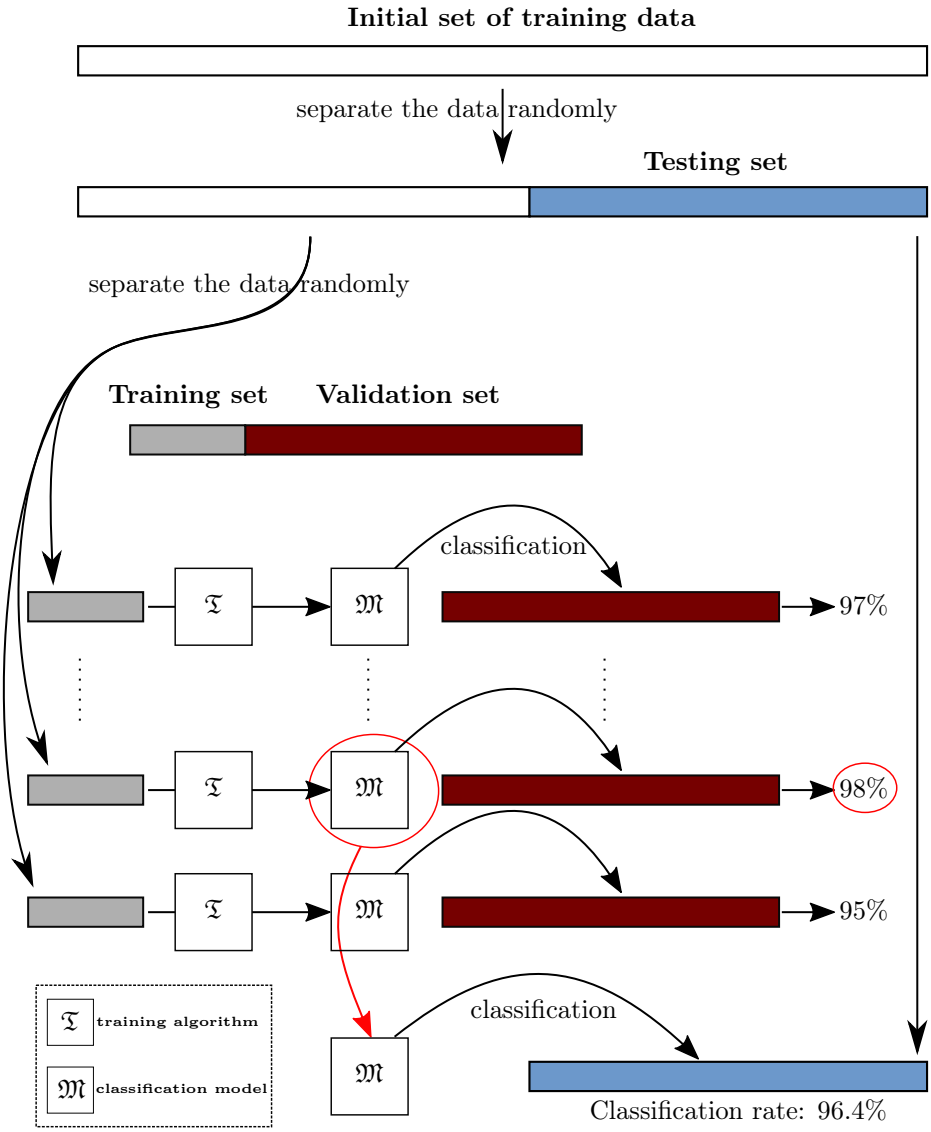


Figure 3.1: A figure presenting the different training phases of machine learning. First, the data is split to isolate a *testing set* (in blue), never to be used during the training process. Then, a number of different models ($\mathfrak{M}$) are created using several randomized sets of *training data* (in gray) with an unchanging training algorithm ($\mathfrak{T}$). The different models are evaluated by classifying the data from the *validation set* (in red). The best model is selected and tested for classification accuracy over the testing test.

where the owner of the model is not the entity performing the inference phase, then either the owner needs to trust an outside entity with their model, or she needs to find alternative ways to ensure confidentiality.

Thus, the cases that have a need for confidentiality-preserving tools are cases where some computation is *outsourced*, whether it be the training phase or the inference phase. We can define our main use-case as such by considering, even in the case of DNA databases, that individual citizens outsource a DNA matching service to the government while still expecting confidentiality for their data.

Essentially, depending on the use-case, there are several desirable confidentiality goals. One could wish to hide: the training data, the nature of the training algorithm, the trained model, the classification data, the classification algorithm or other aspects specific to the given machine learning algorithm (network topology in the case of neural networks for instance).

In this work, we focus on the *inference phase*, with confidentiality properties obtained over the trained model and/or the classification data.

### 3.3.2 Adversaries

We have established that the need for confidentiality-preserving tools is dependent on the outsourced nature of at least part of the computation. Indeed, if both the training and the inference phases happen on the same server, with locally collected training and classification data, then the need for confidentiality can be solved with the means of network security: ensuring there is no information leakage on the server.

In our case, we are therefore outsourcing a computation to an entity. If the entity can be trusted, then achieving confidentiality simply means ensuring that property over the transfer channels used. If, however, this entity cannot be trusted, then there is a need for additional measures. The extent of those measures will depend on the level of trust that we have for them. In the realm of information security, this level of trust is expressed through the notion of *adversary*. The outside entity, if not trusted, is considered an adversary that can be either *semi-honest* or *malicious*. A semi-honest adversary will follow protocol specifications to the letter but will try to learn whatever it can with what it can observe. Malicious adversaries may behave arbitrarily, and not according to the given protocol specifications. In this work, we are considering only the case of a semi-honest adversary.

### 3.3.3 Confidentiality-preserving tools

There are several ways that confidentiality can be achieved when outsourcing some type of machine learning computation. We present them briefly below. First however, we present a tool that is mainly used as a privacy-preserving option: anonymization. The reason for this is the use that it can have, combined with confidentiality-preserving methods, reinforcing the confidentiality of the data.

#### Anonymization

Anonymization, as its name suggests, aims to guarantee user privacy more so than data confidentiality. One way to go about anonymizing a set of records in a dataset would be to simply erase direct identifiers (name, address, social security number, etc..) from the dataset. However, this simple solution simply does not work at all: many successful attacks have been mounted on such anonymized datasets. The most cited one involves the governor of Massachusetts (United States)

assuring his constituents in 1997 that the public release of hospital patient data for the purpose of improving healthcare and reducing costs could not harm them as it had been thoroughly anonymized. An MIT graduate student, Latanya Sweeney, proceeded to identify the governor's own hospital records among the ones released by cross-checking the data with publicly available voter registration records. She sent the governor's health records to his office as a theatrical lesson on anonymization. This example among others makes the case for a set of stronger privacy guarantees for individuals whose records are used in a data analysis protocol.

There are several *privacy models* that define such privacy guarantees anonymization can provide. A user wishing to choose from an anonymization protocol can induce that protocol's security guarantees from its associated privacy model. From the developer of the protocol's point of view, the privacy model is a guideline outlining the minimum security requirements necessary. Since 1997's dramatic lesson on anonymization, there have been two main privacy models introduced (each with a number of variations):

- **$k$-anonymity**. Introduced in 1998 in [SS98], $k$-anonymity states that no user can be linked to less than $k$ dataset records by an adversary. At which point the probability to link a user to its actual record will be at most $\frac{1}{k}$. While $k$-anonymity constrains the protocol with regard to linking a user to a specific record, it does not cover the matter of a specific attribute being linked to a user in the case where some attribute values are widely shared between records. For instance, if the record is a diagnostic summary, then an attribute value for a positive cancer test will take two values: 0 and 1. Therefore, an adversary linking a user to a set of $k$ records all with "cancer" attributes set to 1 will have determined with probability 1 that the user has cancer. The shortcomings of $k$-anonymity led to the development of alternatives: $l$-diversity [Mac+06], $t$-closeness [LLV07] and $n$-confusion [ST12].
- **Differential Privacy (DP)**. DP, as introduced in [Dwo06], states that the result of any computation over a given dataset would not have been different if any individual user had not provided their record. In fact, it gives a framework where that property holds up to a certain probability, controlled by a parameter $\epsilon$. The *fundamental law of information reconstruction* as defined in [DR14], states that "overly accurate answers to too many questions will destroy privacy in a spectacular way". Therefore, a good way to preserve privacy is to introduce noise in the original data: making the overall result less accurate. That is what needs to be done to fit the DP model, all the while ensuring that the noise not be too large, lest the overall result become useless. There are a number of variations on the original DP notion ([DP20] estimates there were around 200 at the time of writing in 2019). This is due to the fact that, for a given use-case where the original DP mandates noises that are too large, one might want to relax the security guarantees to obtain a more accurate result. For such cases, in order to provide potential users with solid security guarantees, one needs to define a new privacy model (albeit only a variation on the original). [DP20] strains to unify these notions under a single conceptual framework for DP variations.

### Cryptography

There are essentially three general methods at a cryptographer's disposal: *MPC*, *FHE* and *FE*.

**Multi-Party Computations.** MPC was first designed by Andrew Yao in the 1980s as a theoretical tool to solve Yao's *millionaire problem*. The problem is that of two millionaires who wish to know which one of them is richer, without learning anything else about each other's

wealth. This problem was only presented as a toy use-case. In general, MPC was designed to implement privacy-preserving applications where multiple, mutually distrusting parties cooperate to compute a function over data shared among themselves. Since the inception of MPC as a theoretical concept, the cryptographic community has developed a vast number of efficient applications, through the means of both *generic* (allowing for any kind of computation) and *specialized* (designed specifically for a given set of computations) protocols. What all of the MPC techniques have in common is a reliance on communication between the two (or more) parties involved as the computation is going on.

**Fully Homomorphic Encryption.** FHE is only slightly older than MPC in its conception. It was imagined in 1978 [RAD78] as a theoretical framework allowing for *public*, *unbounded*, *arbitrary*, computations over encrypted data, yielding an encrypted result. This means three things.

- *Public computation.* Publicly computing something means doing it without access to any secret information: given only $\text{Enc}(x)$, an encryption of a value $x$, and a function $f$, one can compute $\text{Enc}(f(x))$ on their own (non-interactively). This is the main *homomorphic* property that all homomorphic encryption schemes must verify.
- *Arbitrary computation.* Most encryption schemes can be considered *partially homomorphic* in that they allow for *some* types of computations to be run on encrypted data. For instance, the Rivest-Shamir-Adleman (RSA) cryptosystem [RSA78] allows for an unlimited number of modular multiplications and the Paillier cryptosystem [Pai99] for an unlimited number of modular additions. A scheme allowing for arbitrary computations is a scheme that, given any computation $f$, can be parameterized so that we can obtain $\text{Enc}(f(x))$ from $\text{Enc}(x)$.
- *Unbounded computation.* Some encryption schemes can allow for arbitrary functions, but only up to a certain computational depth. This is due to an intentional error introduced in the ciphertext at encryption. This error grows with the number of operations, inducing a *noise propagation* that, after a certain point, leads to an inaccurate decryption. This means that they can only evaluate a certain amount of operations until the result ciphertext is no longer correctly decipherable. These are *Somewhat Homomorphic Encryption (SHE)* schemes. If, for a any given depth $L$, an encryption scheme can be parameterized to compute $L$ operations correctly, then it is called a *Levelled Homomorphic Encryption (LHE)* scheme. A scheme allowing for unbounded computations can be parameterized *a priori*, with a set of parameters that would fit an application over any computation.

Although most agree on what is required for one to be *fully* homomorphic, some of the terms above have been used with different notions in mind in the literature. The definitions we present here seem to be the ones generally accepted nowadays. In any case, they are the ones we are using throughout this thesis.

For just over 30 years after its inception, an implementation of this FHE concept proved elusive. Another way to approach solving the problem was by building *LHE* schemes. In a 2009 article [Gen09b] and his Ph.D. thesis [Gen09a], Gentry proposes the first FHE scheme using lattice-based cryptography with a ground-breaking idea: running the ciphertext into its own decryption process homomorphically, resetting it to its original state and allowing for further computations. This now staple operation in most modern FHE scheme is the *bootstrapping operation*. In the last 11 years, FHE went from a far-fetched theoretical concept, to an increasingly practical and diverse set of frameworks.

**Functional Encryption.** FE was introduced over 10 years ago as an umbrella concept for different encryption goals existing at the time: IBE [Sha84]; searchable encryption [Bon+04]; ABE [SW05]. Although they did not introduced the name itself, [BSW11] formalized the notion for the first time in 2010. It is a generalization of public-key cryptography and has been mostly used in the context of searching on encrypted data or access control. It is a good candidate for more complex applications such as confidential machine learning even though, as of now, its use in this context has not been considered as much as MPC or FHE. Recent work [Ryf+19; Mar+19; Abd+20] has challenged the notion that it is not adapted to this kind of application. However, general-purpose functional encryption such as an easily adaptable framework seems far off. In the context that we presented in Section 3.1, a server needs to compute a function over data from a client without learning anything about this data. FE solves this problem by introducing a third entity: the *authority*. That entity produces a *master secret key* and a *public key*. The client encrypts data using the authority's public key, and sends it to the server. Given a function $f$ that the server needs to compute on the client's data, the authority will derive a secret key $\mathrm{sk}_f$ from its master secret key and send that secret key to the server. The server can then run an evaluation algorithm that outputs the result as a *plaintext*. In short, while FHE evaluates over encrypted data, FE evaluates *and* decrypts over encrypted data.

## 3.4 Notions surrounding cryptography

### 3.4.1 Provable Security

**Perfect secrecy and Semantic security.** Shannon's definition of security is encapsulated in the notion of *perfect secrecy*: when an attacker intercepts a ciphertext, he should be able to learn absolutely no information about the underlying plaintext. Although it would seem ideal that all cryptographic schemes achieve this property, in practice, this would be impractical. Indeed, Shannon's definition assumed that the attacker had access to infinite resources and time. Relaxing Shannon's requirement was the object of Goldwasser and Micali's 1982 work [GM82] introducing the notion of *semantic security*. In [GM84], the authors elaborated on this notion by linking it to the newly defined property of *ciphertext indistinguishability* under a *Chosen Plaintext Attack* (IND-CPA). This definition defines a *game* between an *adversary* and a *challenger*. For a given encryption scheme, if any efficient adversary cannot win the IND-CPA game, then the encryption scheme is said to be *semantically secure*. The game is as follows. The adversary chooses two plaintexts and gives them to the challenger. The challenger sends an encryption of only one of the plaintexts (at random) back to the adversary. The adversary wins if she can determine which one of the plaintexts was encrypted with non-negligible probability.

Harder notions were introduced later: Indistinguishability under a non-adaptive Chosen Ciphertext Attack (IND-CCA1) and Indistinguishability under an adaptive Chosen Ciphertext Attack (IND-CCA2). In the IND-CCA1 game, the adversary has an additional advantage: she has access to an encryption or decryption oracle before the challenge ciphertext is sent. This means she can encrypt or decrypt arbitrary messages at will. In the IND-CCA2 game, the adversary has access to this oracle even after the challenge ciphertext is sent, but cannot asks for decryption of the challenge ciphertext.

In practice, the time it takes for the adversary to win the game will define the security of the scheme. More precisely, for a given integer $\lambda$, the adversary must not be able to win in $\mathcal{O}(2^\lambda)$ operations. We call $\lambda$ the security parameter and keep this notation throughout this work.

**Security of FHE schemes.** These three games define a growing level of semantic security: any scheme that is IND-CCA2 secure is IND-CCA1 and IND-CPA secure and so on. There are other notions defined by other games, but these three interest us most because they allow us to define precisely where we will stand in terms of security standards in this thesis.

First, by construction, an additive homomorphic scheme cannot be IND-CCA2 secure. Indeed, given the challenge ciphertext, the adversary can ask the oracle for an encryption of any message $m$. Then, she adds the resulting ciphertext to the challenge ciphertext, asks for a decryption of that ciphertext to the oracle and subtract $m$. This will give her a decryption of the challenge ciphertext due to the scheme's additive homomorphic property. IND-CCA1 security, on the other hand, is not in natural contradiction with the homomorphic property. There are homomorphic schemes that are IND-CCA1 secure [CS98; Can+17]. However, any scheme using Gentry's bootstrapping idea [Gen09b] cannot be IND-CCA1 secure by construction: the bootstrapping key is an encryption of the secret key shared publicly. Therefore, a simple call to the decryption oracle will give the adversary knowledge of the entire secret key. In this thesis, we make and extensive use of the TFHE encryption scheme, which is semantically secure (IND-CPA) but which uses a bootstrapping operation and therefore is not IND-CCA1 secure.

**Hardness assumptions.** Depending on the encryption scheme, the evaluation of its security (the time it takes for an efficient adversary to win the game) will depend on one or more mathematical problems. The initial game winning condition is *reduced* to given mathematical problems to make a precise evaluation of the strength of the adversary's attack. For instance, a problem can be factoring large composite numbers, or finding the shortest vector in a lattice. Once the mathematical problem has been defined, the fastest known algorithm given the rules of the game is assumed to be used by the adversary. Obviously, what interest cryptographers are "hard" problems that will take an impractical amount of time for any adversary to solve. Obtaining a definitive proof that such a problem is actually "hard" is a whole other matter. One that we could not possibly delve into with great detail, as it was not the subject of our research.

### 3.4.2 Fully Homomorphic Encryption

**An overview of past encryption schemes** Because we use FHE extensively in our work, we present here a brief history of the subject matter. This is not solely for the purpose of giving recognition to great previous work on the basis of which this work is built: it gives a better idea of where this work is coming from and thus where it can lead.

FHE, from Gentry's founding work [Gen09a; Gen09b], is traditionally seen as having gone through 3 phases: the 3 generations of FHE. These give a better understanding of the challenges that this field has overcome, and the ones that still stand in the way of practical real-world applications.

- *The first generation.* The first generation is directly based on Gentry's scheme: a public-key encryption scheme over ideal lattices in a polynomial ring [Gen09a; Gen09b]. The scheme is "straightforward" in the sense that homomorphic additions and multiplications correspond to additions and multiplications in the polynomial ring. Then, all of the potential homomorphic encryption schemes were based on noisy encryption: encrypted data contains an error that is needed for the security of the scheme. All the FHE schemes we know today still require this noise. A crucial security requirement becomes a liability in an homomorphic setting: with every operation the noise grows, until the ciphertext can no longer be decrypted correctly. Gentry's groundbreaking idea was the use of a novel

operation: *bootstrapping*. The idea is to apply a re-encryption and then decryption procedure homomorphically. This resets the noise and allows for more operations to take place. Once the noise hurdle was overcome, the door to more efficient and practical FHE was open. Where problems still lied at the time, was in the complexity of the novel *squashing* technique that simplified the decryption function under the security assumption that the Sparse SubSet Sum problem (SSSS) be hard. A first - partial - implementation [SV10] did not succeed in implementing the squashing and therefore achieved only an LHE scheme. Gentry later implemented his own scheme along with Halevi in [GH11]. At the time, a single bootstrapping operation would take from 30 seconds to 30 minutes depending on the parameters. The main goal moving forward for researchers in the field would be to design and implement faster schemes, and ones not relying on the SSSS assumption.

- *The second generation.* The second generation of FHE saw the light with the works of the Brakerski and Vaikuntanathan in the early 2010s. In [BV11a; BV11b], they built an efficient LHE scheme (the BV scheme) based on Learning With Error (LWE) worst-case hardness assumptions (see Chapter 5 for definitions). One drawback of the initial scheme was the prohibitive size that parameters would have to grow to allow for high multiplicative depths. This is a problem for any levelled homomorphic encryption scheme. [BGV12] introduced a novel *modulus switching* technique. When associated with the tensoring technique introduced in [MGH10], it allows for a slower noise growth and therefore smaller parameters for a given multiplicative depth This in turn makes the scheme much more efficient. The BGV encryption scheme is a reference for second generation FHE. In [Bra12], Brakerski improves on previous schemes by reducing the noise propagation and [JF12] takes Brakerski's scheme and ports it to a Ring-LWE setting (see Chapter 5 for definitions) to create the B/FV scheme.
- *The third generation.* In [GSW13], Gentry *et al.*introduce a new scheme that does not rely on the costly re-linearization step that previous schemes - GSW - used to keep the noise low. [BV14] introduce a bootstrapping operation which is then improved upon in [AP14]. GSW can be seen as the matrix equivalent to LWE. Both FHEW [DM15] and TFHE [Chi+16a] are later schemes built upon GSW and its fast bootstrapping operation.

### 3.4.3 Cryptography in outsourced computations

We recall that, in the end, our goal is to use FHE as a confidentiality-preserving tool for outsourced machine learning algorithms. There are several ways that this could work, but we present in Figure 3.2 a possible protocol. In it, a "query" $q$ is sent encrypted by a "user" to a remote server hosting a machine learning algorithm. The algorithm is run homomorphically over that encrypted input and the result $r$ is sent back to the user for decryption.

It is quite a simple protocol and yet on the surface very effective in ensuring confidentiality for both the user query, which is encrypted during the whole process, and - at first glance - the machine learning model, which stays on its dedicated server. In reality, while on the side of the user this protocol is semantically secure, there are certain security properties that it does not achieve naturally.

**Verifiable computation.** Among those, we mention that the user has no guarantees for the validity of the result she receives. What is called *verifiable computation* (a notion introduced in 2010 by Gennaro *et al.*[GGP10]) is no small issue when it comes to non-interactive outsourced computations and is the subject of its own set of works. Specifically, modern verifiable computation over homomorphic computations is limited by the complexity (precisely, the degree of

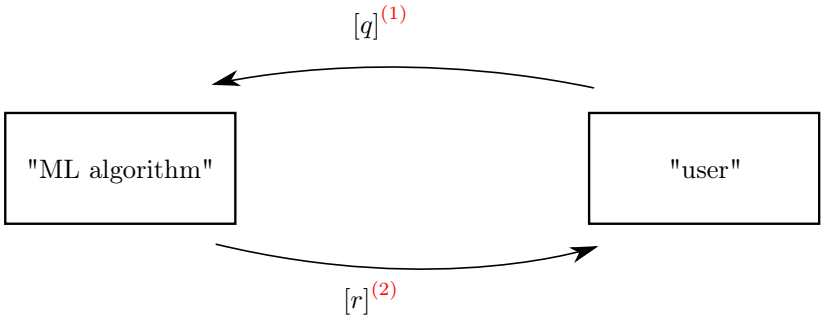


Figure 3.2: A general possible protocol for FHE usage in a confidentiality-preserving outsourced machine learning scenario. We use brackets ($[\cdot]$) to indicate that data is encrypted. The order in which the data is exchanged is indicated in red: $q^{(2)}$ means $q$ is the second data exchange to happen in the protocol. A "query" $q$ is sent encrypted by a "user" to a remote server hosting a machine learning algorithm. The algorithm is run over that encrypted input and the result $r$ is sent back to the user for decryption.

the polynomial) of the given computation. To our knowledge, there is no result that allows for homomorphic verifiable computation over anything more complex than multivariate quadratic polynomials. We mention [FGP14] as a well-recognized work on homomorphic verifiable computation, recently improved upon in [FNP20]. Although verifiable computation seems still far from widespread practical use, machine learning applications have started to appear in recent works (see [KMC18; MS20]). For a relatively modern survey on the matter, see [DSB17].

**Model confidentiality.** There is another issue with the protocol illustrated in Figure 3.2: with enough requests, the user may be able to learn quite a lot about the machine learning model stored on its dedicated server, therefore compromising confidentiality. There is actually no provable security on the model data. This very problem is the matter addressed in [VJ10], whose title speaks for itself: "On the Impossibility of Cryptography Alone for Privacy-preserving Cloud Computing". Controlling what the user can learn about the model can be, sometimes, crucial to ensure that the use of FHE yields the desired properties. There are several methods that can be used to complement FHE as answers to this problem. Noising the output of the classifier (differential privacy gives us the theoretical toolbox to do so meaningfully) is one. This countermeasure is hard to consider in serious use-cases such as medical ones, where patient health - and therefore classification accuracy - cannot be traded for database confidentiality. Another possible countermeasure would be throttling the request rate, which can be performed by a given entity. In any case, this is a problem that all secure outsourced computation protocols need to address, and is not specific to FHE. We do not go into any more detail on the matter in this work and view the search for solutions to this issue as a separate line of research.

**Circuit privacy.** Apart from what the user can learn from the result that she obtains from the server, one could wish to hide any more information that could be obtained not from the plaintext result, but from the result ciphertext sent by the server. In the literature, this property is referred to as *circuit privacy*. The information leak comes a priori, from the noise in the output ciphertext. The user can measure this noise and learn information about the function applied on her data. There are several ways to achieve this property.

In Gentry's thesis [Gen09a], the server adds a noise to the output ciphertext before sending

| Setup | (1) | (2) | (3) | (4) | (5) | (6) | (7) |
|---|---|---|---|---|---|---|---|
| The client sends its query in clear form. The server applies the computation and sends the result to the client. | ∼ | ∼ | ∼ | ✗ | ✗ | ∼ | ✗ |
| The server sends the data to the client in clear form. The client applies the computation itself. | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ |
| The server sends the (perfectly anonymized) data to the client who applies the computation itself. | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |
| The client sends its query encrypted homomorphically. The server applies the computation and sends the result to the client for decryption. | ∼ | ∼ | ∼ | ✓ | ✓ | ∼ | ✓ |

Table 3.2: This table is an update of Table 3.1, with the addition of a basic use of FHE on the last line. The ∼ symbol represents a property that is not achieved because of information leakage, while the ✗ symbol means the property is trivially not achieved because data is sent in clear form.

it to the user. If the noise is high enough, it will drown out the information that the user could retrieve. This method requires an increase in the size of the parameters used and therefore reduces the scheme's efficiency. An alternative is introduced in [Bou+16], where the authors show that adding a small noise at every operation during the homomorphic computation is enough to achieve circuit privacy with a low impact on efficiency.

Another method is the use of a bootstrapping operation which resembles a decryption/re-encryption operation. Ideally, this could therefore erase any trace of past homomorphic operations. However, the output noise distribution for the bootstrapping operation is not *canonical*: it does not correspond to the noise distribution of a freshly encrypted ciphertext. [DS16] propose a solution based on several rounds of bootstrapping operations with small noise additions in between.

Although both of these solutions could be applicable in the schemes that we present, we do not evaluate their potential impact and do not consider circuit privacy.

### 3.4.4 Confidentiality and Privacy with FHE

We recall that we presented an example of outsourced computation in Section 3.1. For that example, we defined a set of privacy and confidentiality properties that a given protocol could or could not achieve. We defined, and numbered, 7 such properties: *record confidentiality* (1); *user privacy* (2); *attribute privacy* (3); *query confidentiality* (4); *client privacy* (5); *computation confidentiality* (6); *result confidentiality* (7). We propose an updated Table of what protocols achieve which properties in Table 3.2 in light of the security considerations presented in Section 3.4.3.

Properties $\{1, 2, 3, 6\}$ correspond to privacy and confidentiality on the server side. In our scenario (the last line of the table), these properties can never be fully achieved without compromising on the accuracy of the result by using differential privacy for instance, and limiting the performance of the scheme by applying a circuit privacy method. There are a number of tools at a cryptographer's disposal, and in the end it comes down to a matter of priorities. In this thesis, we only explore "simple" schemes that do not integrate either anonymization methods or circuit privacy methods.

# Chapter 4

# State of the art

**Contents**



## 4.1 A time before FHE

Up to the turn of the millennium, the main secure techniques were not envisioned for the kind of data-hungry applications we know today. Interest for secure data mining algorithm grew in tandem with the development of the World Wide Web through the 90s and into 2000s. Two works [LP00; AS00] can summarize the state of research in the field at the turn of the millennium. Both were published in the year 2000 and are named *Privacy-Preserving Data Mining*. [LP00] uses an MPC solution to the privacy problem posed by data-mining applications. [AS00] focuses on confidentiality and uses anonymization techniques available at the time.

In the following, we present the progress that has been made since on the front of *privacy-preserving* machine learning *evaluation*. Indeed, we restrict our presentation to works solving the privacy problem, and therefore leave anonymization-based solutions (those that ensure confidentiality) aside. Additionally, even though the question of achieving privacy of training data is a natural perspective for the work that we present, it is not directly related to it.

Most of the progress on the privacy-preserving evaluation problem in the decade that followed [LP00] consisted of pushing the accuracy and efficiency of MPC-based schemes even further. These schemes consist of a combination of Additive Homomorphic Encryption and tools from the MPC toolbox. With the appearance of practical fully homomorphic schemes, a parallel research direction was opened: the search for FHE-only solutions. We recall that while MPC and FHE provide similar services in theory, FHE does so non-interactively. Therefore, the interest in eliminating the need for MPC stems from use-cases where network latencies and high bandwidth usage are common issues.

## 4.2 Neural Network Evaluation

In 2009, FHE-based schemes came into their theoretical existence. It was not until 2014 that the first practical method for non-interactive homomorphic classification of a neural network - Cryptonets - was presented in [Xie+14] and then published as [Dow+16]. The team responsible for this work later kept developing their project. Their latest development is presented in [BEG19]. The main idea was to use *levelled* homomorphic encryption on an approximate neural network (the activation function was approximated with the square function). Their use of LHE means restrictions regarding the depth of the neural network. Indeed, neural networks are usually structured in consecutive layers. The number of layers defines the depth of the network. In theory, this is not an issue. Indeed, one can simply set the parameters of their scheme to match the given neural network depth. In practice, this poses several issues. On the one hand, an entity performing an encryption operation must know the encryption parameters. If that entity encrypts data for classification on a distant - hidden - neural network, then they will learn something about the neural network from the parameters they are given to perform the encryption. In this case part of the topology of the network is leaked, although that is not a grave matter. On the second hand, handling a "deeper" network with this method means increasing the size of the overall parameters. This, in turn, increases the time every operation will take to complete at every layer. As a general rule, if a given machine learning algorithm has a linear complexity with regard to the size of its topology, then its *levelled* homomorphic equivalent will have a higher complexity, linked with the rate of parameter growth. All of this is not to diminish the impact of such LHE-based schemes, but rather give some perspective surrounding their limits. It is also meant to present the interest in developing FHE-based schemes, which do not present these limitations. Without going into details, we cite here several recent works that improved on Cryptonets while following its general approach: [ZYC16; Cha+17; Cha+19]. All of these works have significantly improved Cryptonets' accuracy, efficiency, and scaling resilience, making a strong case for practical secure machine learning. Lastly, work is now being produced to make homomorphic integration seamless to the eyes of the user, with the goal to make homomorphic encryption just another option in a wider machine learning framework. Cingulata [CDS15] and nGraph-HE [Boe+19] are such frameworks.

The first work to present an efficient, *fully*-homomorphic, non-interactive evaluation of a neural network was published in 2018 [Bou+18]. It uses a very efficient bootstrapping algorithm introduced in [Chi+16a]. The square function used in [Dow+16] is replaced by a sign function computed by the bootstrapping operation.

While much progress has been made towards homomorphic-based schemes in this area of research, it is important to note that MPC-based schemes remain much faster. This fact does not invalidate past and present research (such as our work) on FHE. For one, FHE is a much more recent research area in which progress has, as of yet, been quick to come. Additionally, FHE does not suffer from network latencies and high bandwidth usage problems, and therefore addresses use-cases that MPC cannot. In any case, FHE is worth investigating in its own right. For reference, the application of other MPC for computing over encrypted data to neural networks also start to be investigated in their associated communities (e.g., [Bal+19], [RRK17]).

## 4.3 Other Machine Learning algorithms

Neural networks, feed-forward neural networks to be precise, are the prodigy children of Machine Learning. Their use and development over the last decade has pushed farther than many would have predicted the strengths of modern "artificial intelligence", and its reach in all corners of

society. However, neural networks are not the only machine learning algorithms used routinely today. Additionally, their complexity makes them particularly challenging for the design of efficient secure alternatives. Therefore, a worthy line of research consists of finding alternative ways to procure privacy-preserving machine learning services.

We presented the fundamental problem that this work inspired by giving an example of generalized DNA-matching use in China, and its progressive spread in the Western hemisphere. The interest in the subject is great, with several solutions proposed over the years. Many of these have been developed at least in part for participation in the iDASH Privacy & Security challenge for secure genome analysis[1]. The specific goal of this challenges is to help develop homomorphic encryption-based methods for the DNA-matching problem (with possible complementary MPC techniques). For context, this problem consists of a hamming distance computation and a comparison. Several works have been proposed in the last decade. Most of them use homomorphic encryption for the distance computation but resort to MPC for a comparison. Indeed, the comparison phase is the most complex to achieve without interaction. To give a single example, [Ash+18] won the iDASH challenge with such a scheme in 2018. A non-interactive, *levelled* homomorphic solution was first proposed by [CKL15]. They achieved this using an homomorphic Boolean circuit. The same method was used by the 2019 iDASH challenge winners (the TFHE-Chimera team, see [Bou+20] for their underlying scheme), but with a *fully* homomorphic version. This method allows for an unlimited amount of computations and makes comparisons very easy at significant efficiency costs: homomorphic bit-wise operations are much costlier than "standard" homomorphic operations such as additions and multiplications on the whole values.

We will not go into this much detail about every possible secure machine learning application. The above is worth mentioning only because the development and state of homomorphic-based schemes in other applications strongly mimic what we have presented about genomics. Essentially, a wide array of applications require a scalar product computation (the hamming distance computation in the genomics case), and an analysis phase consisting of the application of a complex non-linear function (the comparison phase in the genomics case). Homomorphic scalar products are accessible to modern homomorphic schemes, however that is not the case for complex non-linear functions. The solutions then are the following:

- The analysis phase is made interactively. This is the case in an overwhelming majority of the works in the literature. It requires the use of tools from the MPC toolbox.
- The complex non-linear function is simplified. This is the case in both [Dow+16] and [Bou+18]. Although this method can be used to very little accuracy loss in the case of neural networks among other applications, its use may sometime incur information leakage (when a square function is used instead of a thresholding function at the output of an algorithm for instance), or may simply not be appropriate given the algorithm's necessities.
- An homomorphic Boolean circuit is used. This solution fits all problems in theory and is therefore very strong. However, it is also usually slower than alternative methods.

## 4.4 Our contributions

The work that we present in this thesis follows the path [Dow+16] and [Bou+18] started on. Our contributions are all fully non-interactive. In our first contribution (Chapter 6), we present an homomorphic recursive neural network where, as in [Bou+18], the activation function is replaced

[1] http://www.humangenomeprivacy.org/

with the sign function. In our second contribution (Chapter 7), we looked at a complex machine learning algorithm called an embedding-based neural network. One of its key properties for us is the separation between the CNNs that are trained once and for all, and the classification model that is created using those CNNs and used for the classification phase. We cannot afford to provide confidentiality during the evaluation phase of a CNNs (here the model creation phase), however we provide two novel homomorphic argmin algorithms that allow us to protect the model during the classification phase. In our third contribution (Chapter 8), we looked at one of the bedrocks of classic machine learning algorithms: the $k$-NN classifier. We provide the only (to our knowledge) non-interactive confidentiality-preserving $k$-NN classifier in the literature by building on one of our previously imagined argmin algorithms. The strengths of the $k$-NN classifier lie both in its relative simplicity with respect to the more modern machine learning algorithms and in its wide modern-day usage in an array of applications.

# Chapter 5

# Technical preliminaries

**Contents**



In this section, we present a host of precise details about the works that we build our research on, as well as the notations that we choose, which might differ from the notations in these same works, but allow us to present a unified vision of our research to the readers.

## 5.1 General notations

**Sets.** Given two integers $a, b$, $a \leq b$, $\{a, b\}$ denotes the set of all integers between (and including) $a$ and $b$. Given two real numbers $x, y$, $x \leq y$, $[x, y]$ denotes the set of all real values between (and including) $a$ and $b$.

For standard mathematical sets, we use the standard notations: $\mathbb{B} = \{0, 1\}$; $\mathbb{N}$ is the set of the unsigned integers (hereby simply known as "integers"); $\mathbb{Z}$ is the set of the signed integers; and $\mathbb{R}$ is the set of all the real values.

$\mathbb{T}$ - "the torus" - denotes the Abelian group $([0, 1], +)$.

$\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denotes the signed integers $\mod q$ with $q$ a given integer.

The cardinal of a finite set $A$ is written $|A|$.

We write $\|\cdot\|_p$ to denote the $\ell_p$ norm of vectors over reals or integers.

Vectors can be noted both with the notation $\vec{*}$ or with no notation at all. The notation $\vec{*}$ is exclusive to vectors. Every variable is clearly introduced as either a vector or a scalar value and the choice in notation is made for the purposes of clarity when necessary.

A scalar product between two vectors $\vec{a}$ and $\vec{b}$ is denoted $\vec{a} \cdot \vec{b}$.

**Polynomials.** The ring $\mathbb{Z}[X]/(X^N + 1)$, where $N$ is a fixed integer, is denoted $\mathfrak{R}$ and the ring $\mathbb{Z}_q[X]/(X^N + 1)$, $\mathfrak{R}_q$. $\mathbb{B}_N[X]$ is the subset of $\mathfrak{R}$ composed of the polynomials with binary coefficients.

We write $\mathbb{T}_N[X]$ the quotient $\mathbb{R}[X]/(X^N + 1) \mod 1$, with $N$ a given integer.

With $\zeta_n = e^{2\pi i/n}$, the $n^{\text{th}}$ *cyclotomic polynomial* is defined by

$$\Phi_n(x) = \prod_{\substack{1 \leq a < n \\ \gcd(a,n)=1}} (x - \zeta_n^a)$$

**Probability distributions.** If $\mathcal{D}$ is a probability distribution over a set $A$, we write $x \xleftarrow{\mathcal{D}} A$ to denote that $x$ is sampled according to $\mathcal{D}$. When sampling $x$ from a set $A$ uniformly at random, we write $x \xleftarrow{\$} A$.

The Gaussian distribution over $\mathbb{R}$ centered at 0 and of *standard deviation* $\sigma \in \mathbb{R}^+$ (denoted $\mathcal{N}(0, \sigma)$) is defined as a distribution having a density function equal to

$$\rho_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}$$

The *width parameter* of a Gaussian distribution is defined as $\sqrt{2\pi}\sigma$.

In the rest of the thesis, when not specified otherwise, we use a Gaussian distribution when sampling randomly.

**Encryption** All of the encryption schemes that we use have the same general characteristics in that they all:

- can encrypt either a scalar value or a polynomial.
- use a noise parameter (e.g. $\alpha$) in their encryption algorithm
- use a secret key (e.g. $\vec{s}$) in their encryption algorithm

Therefore, whatever the underlying encryption scheme, the ciphertext of a scalar value $\mu$ (resp. a polynomial $\mu[X]$), encrypted using the noise parameter $\alpha$ and the key $\vec{s}$ is written $[\mu]_{\vec{s},\sigma}$ (resp. $[\mu[X]]^{(r)}_{\vec{s},\alpha}$). Both $\vec{s}$ and $\alpha$ can be omitted from the notation when their addition is not integral to the understanding of the reader and when their removal helps with clarity.

## 5.2 Learning With Error

**The LWE problem.** The LWE problem was introduced by Regev [Reg05] in 2005, and it has two variants: Decision-LWE and Search-LWE. They are defined as follows.

We let $n, q \in \mathbb{N}$ be positive integers and $\chi$ be a noise probability distribution over $\mathbb{Z}$. Let $\vec{s} \in \mathbb{Z}_q^n$. Then let $\mathcal{D}$ be the probability distribution defined by choosing $\vec{a} \xleftarrow{\$} \mathbb{Z}_q^n$, then $e \xleftarrow{\chi} \mathbb{Z}$ and considering it in $\mathbb{Z}_q$, and returning:

$$(\vec{a}, b) = (\vec{a}, \vec{a} \cdot \vec{s} + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

*Decision-LWE* is the problem of determining whether a given $(\vec{a}, b)$ was taken from $\mathcal{D}$ or from $\mathbb{Z}_q^n \times \mathbb{Z}_q$ uniformly at random.

*Search-LWE* is the problem of recovering the key $\vec{s}$ from a given $(\vec{a}, b) \xleftarrow{\mathcal{D}} \mathbb{Z}_q^n \times \mathbb{Z}_q$.

In fact, Decision-LWE and Search-LWE are equivalent problems and have been proven hard when assuming the hardness of certain lattice problems. This makes LWE a good problem to base cryptographic schemes on. That is what Regev did in [Reg05].

In general, the distribution $\chi$ is considered to be a discrete Gaussian distribution centered on 0 and with *width parameter* $\alpha q$ with a given $\alpha$. This means its *standard deviation* is $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$. We can write $\chi = \mathcal{N}(0, \frac{\alpha q}{\sqrt{2\pi}})$. The parameters that entirely determine the LWE problem are therefore $n$, $q$ and $\alpha$. $\alpha$ is the *noise parameter*.

**The Ring-LWE problem** Starting in 2010, Lyubashevsky *et al.*[LPR10; LPR12] introduce a variant to the LWE problem: Ring-LWE (RLWE). The RLWE problem is structured in exactly the same way as LWE. First we define a distribution.

We let $\mathfrak{R} = \mathbb{Z}[X]/f(X)$ be a polynomial ring with $f$ a cyclotomic polynomial of degree $N$. In our case we always have $f(X) = X^N + 1$ with $N$ a power of 2. Let $q \geq 2$ be a positive integer modulus and $\mathfrak{R}_q = \mathfrak{R}/q\mathfrak{R}$. Let $\chi$ be a noise probability distribution over $\mathfrak{R}$. Choose $\vec{s} \in \mathfrak{R}_q$. Then let $\mathcal{D}$ be the probability distribution defined by choosing $\vec{a} \xleftarrow{\$} \mathfrak{R}_q$, then $e \xleftarrow{\chi} \mathfrak{R}$ and returning:

$$(\vec{a}, b) = (\vec{a}, \vec{a} \cdot \vec{s} + e) \in \mathfrak{R}_q \times \mathfrak{R}_q$$

*Decision-RLWE* is the problem of determining whether a given $(\vec{a}, b)$ was taken from $\mathcal{D}$ or from $\mathfrak{R}_q \times \mathfrak{R}_q$ uniformly at random.

*Search-RLWE* is the problem of recovering the key $\vec{s}$ from a given $(\vec{a}, b) \xleftarrow{\mathcal{D}} \mathfrak{R}_q \times \mathfrak{R}_q$.

As with LWE, Decision-RLWE and Search-RLWE are equivalent problems.

**(R)LWE-based encryption schemes.** For a precise description of an encryption scheme based on the LWE or RLWE problem, we refer to Section 5.3, where we present the TFHE encryption scheme. Importantly, all (R)LWE-based homomorphic encryption schemes work by introducing an error into the ciphertext. When the ciphertext is added or multiplied to other ciphertexts, that error will grow, inducing an *error propagation*. The larger the noise, the lower the probability of correct decryption. RLWE-based encryption is more efficient than its LWE

counterpart because of its compactness (each $b$ is $N$-dimensional) and because of implementation optimizations such as the use of a Fast-Fourier-Transform.

**The security of LWE-based encryption schemes** The security of LWE-based schemes depends directly on the strength of the attacks that can be mounted against them and thus evolves over time. The discovery of faster attacks usually means that the parameters used for an implementation of an LWE-based scheme must be changed, at a performance cost. These parameters are set according to a desired security requirement: the security parameter $\lambda$ as defined in Chapter 3.

An important work was provided in [APS15] in estimating the hardness of LWE. From that work was born the LWE estimator[1], a Sage module providing functions for estimating the hardness of LWE. Given a set of parameters, it summarizes existing attacks on LWE and provides a minimum security parameter $\lambda$ that gives a clear estimation of the security of the scheme implemented with those parameters.

The results in [APS15] are now obsolete and more recent results can be found in [Pla18]. The LWE estimator is kept up-to-date with the latest development in the field of cryptanalysis and is therefore the best source of security requirements for one wishing to parameterize LWE-based encryption schemes.

However, it is important to note that schemes implemented using the LWE estimator will probably need to keep their parameters constantly up-to-date with the latest cryptanalysis developments. The ease with which one can do this is the strength of the LWE estimator, and its results should not be considered as a definitive proof of security in the long term. Since all of our work is based on the LWE problem, this means that our performance results are subject to change in time, as parameters evolve and adapt to new security standards. This will affect the performance of what we present, but not the overall validity of the algorithms we introduce. Since their inception, all LWE-based schemes have been affected by this instability. Ideally, little by little, security standards will stabilize for them in the future, making it easier for everyone to implement them.

The estimator is conservative in its estimations in the sense that it assumes the adversary to have access to an optimal (from the point of view of the attacker) number of LWE samples $(\vec{a}, b)$ to solve LWE. This is a good practice for most use-cases. However it also provides (thanks to [Bin+18]) an extension that enables the user to choose a specific number of samples to give the adversary access to. Restricting the adversary in this way can best represent the situation in some cases and, when it does, it allows users to choose more efficient parameters. *We base the security of all of the schemes presented in this thesis on the LWE estimator results, placing ourselves in the general case where the adversary has access to an unlimited amount of samples.* The focus of this thesis is not to address cryptanalysis questions relating to the security of various schemes.

**The security of RLWE-based encryption schemes** It is generally assumed that all versions of the RLWE problem can be reduced to their equivalent version of the LWE problem. If this is true, it makes RLWE-based encryption schemes as secure as LWE-based encryption schemes. However, at the time of writing this thesis, no such proof exists. The problem is still widely used as a basis for encryption schemes because of the great advantages it offers to cryptographers: smaller keys and faster operations.

[1] https://bitbucket.org/malb/lwe-estimator/

## 5.3 The TFHE encryption scheme

The TFHE encryption scheme was proposed in 2016 [Chi+16a], as a novel FHE scheme based on the TLWE problem that the authors introduced. It was improved in [Chi+17] and both works were recently unified in [Chi+19]. The TFHE scheme is implemented as the TFHE library [Chi+16b].

### 5.3.1 Torus-LWE

In 2016, Chillotti *et al.*[Chi+16a] introduce the Torus-LWE (TLWE) problem, which they generalize in [Chi+19] as an "abstract TLWE problem" aiming to unify every scale-invariant FHE scheme. Their definition of this new abstract problem through the lens of a set of *phase* functions allows them to present a general problem that can be reduced to other problems by choosing the appropriate phase. For the purposes of clarity, we will present here their "canonical" declination: the last LWE variant needed to explain our results. Here is how they define what we will now call the *TLWE problem*:

Let $k \in \mathbb{N}^*$, $N$ be a power of 2 and $\sigma \in \mathbb{R}^+$ a standard deviation. Choose $\vec{s} \xleftarrow{\$} \mathbb{B}_N[X]^k$ and $\vec{a} \xleftarrow{\$} \mathbb{T}_N[X]^k$ uniformly at random. Choose $e \xleftarrow{\mathcal{N}(0,\sigma)} \mathbb{T}_N[X]$. At this point, return:

$$(\vec{a}, b) = (\vec{a}, \vec{a} \cdot \vec{s} + e) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$$

When $k = 1$, the TLWE becomes the torus equivalent of the RLWE problem. When $N = 1$, the TLWE becomes the torus equivalent of the LWE problem with $k = N$. Additionally, we can define a *noise parameter* $\alpha$ in this case as well: $\alpha = \sqrt{2\pi}\sigma$. The fact that we have two definitions for $\alpha$ is not an issue: standard LWE-based schemes such as B/FV and TLWE-based schemes are unified under a single framework: Chimera [Bou+20]. Under that framework, as we present in Section 5.4, these two $\alpha$ definitions are found to correspond to the same value.

### 5.3.2 TRGSW encryption

The TFHE encryption scheme is built on the GSW [GSW13; AP14] scheme. Indeed, [Chi+16a] introduce the TRGSW encryption, which is very similar to GSW. The main difference is the fact that TRGSW is approximate while GSW is precise. This fact, however, does not accurately depict the reality of both schemes. Both GSW and TRGSW use a *gadget decomposition* to represent their message. What this means for GSW is the use of a binary decomposition of the original message. Since that scheme restricts its message space with a modulus $p$ it can therefore reach a perfect precision on its binary decomposition by using $\ell = \lceil \log q \rceil$ slots. TRGSW on the other hand, does not restrict its message space. It therefore cannot preset a value for $\ell$. On top of that, TRGSW allows for a non-binary decomposition: using a base $B_g$ to be set as a parameter of the scheme. We only mention all of this to make sense of the parameters $\ell$ and $B_g$ that we are mentioning later in the manuscript. We do not go into more detail for either GSW or TRGSW and refer to the cited papers for reference.

### 5.3.3 T(R)LWE encryption

The TLWE and TRLWE encryption and decryption algorithms are one and the same under the unified work in [Chi+19]:

**T(R)LWE encryption algorithm** Let $k \in \mathbb{N}^*$, $N$ be a power of 2 and $\sigma \in \mathbb{R}^+$ a standard deviation. Choose $\vec{s} \xleftarrow{\$} \mathbb{B}_N[X]^k$ as an encryption key. Let $\mathcal{M} \subset \mathbb{T}_N[X]^k$ be the discrete message space. To encrypt a given message $\mu \in \mathcal{M}$ under the key $\vec{s}$, sample $\vec{a} \xleftarrow{\$} \mathbb{T}_N[X]^k$ uniformly at random and $e \xleftarrow{\mathcal{N}(0,\sigma)} \mathbb{T}_N[X]$. Then, compute

$$b = \vec{s} \cdot \vec{a} + \mu + e \in \mathbb{T}_N[X]$$

$(\vec{a}, b)$ is called a *fresh* ciphertext of $\mu$. Notably, any fresh ciphertext of any message $\mu$ can be obtained by summing a ciphertext of the message 0 - $(\vec{a}, b)$ - with the plaintext $(\vec{0}, \mu)$. Any such ciphertext of 0 can then be seen as a public encryption key in a public encryption scheme based on this private encryption scheme. A ciphertext is no longer fresh when it is obtained through homomorphic operations with other ciphertexts.

**T(R)LWE decryption algorithm** Decryption is made by computing the *phase* function $\phi_s$: $\phi_s(\vec{a}, b) = b - \vec{s} \cdot \vec{a}$. We then round $\phi_s(\vec{a}, b)$ to the nearest element in $\mathcal{M}$. In the case of a fresh ciphertext for instance, we have $\phi_s(\vec{a}, b) = \mu + e$. Therefore, if the error $e$ was chosen to be small enough (and yet high enough to ensure security) then the decryption will be accurate.

**TLWE and TRLWE encryptions** In the encryption algorithm above, setting $k = 1$ means using the "polynomial" TRLWE encryption, and setting $N = 1$ means using the "scalar" TLWE encryption. The decryption algorithm does not change. In practice, to fit with the tradition in other works using TFHE, when using the TLWE encryption, we use the notation $n$ instead of $k$. Therefore, $k$ will always be set to 1 in our work.

### 5.3.4 TLWE, TRLWE and TRGSW

One of the strengths of TFHE is in the interactions between these three methods of encryption. Essentially, TRGSW encrypts integer polynomials, TRLWE torus polynomials and TLWE torus scalar values. On top of this, one can encrypt three messages in these three encryption schemes using the same key (up to some re-writings). Indeed, a TRGSW is a matrix of TRLWE ciphertexts, while scalar coefficients can be extracted from TRLWE ciphertexts into TLWE ciphertexts. Additionally, with a Circuit Bootstrapping operation, a TRLWE ciphertext can be "lifted" to a TRGSW ciphertext given a modulus with which to rescale the torus values into integer values. We present more details on these operations (extraction, circuit bootstrapping) in Section 5.3.8. Other similar cross-encryption operations such as the public functional key-switching (TLWE to TRLWE) are not used in our work and therefore are not presented.

While there are 3 distinct encryption schemes (TLWE, TRLWE, TRGSW), for clarity we do not usually specify which encryption scheme we are speaking of when we mention a ciphertext. The reason for this is that we usually assume that ciphertext to be a TLWE or a TRLWE ciphertext and the distinction is then made according to whether we encrypt a scalar value or a polynomial value as we describe it in Section 5.1. With TLWE, we encrypt scalar values and use the simple bracket encryption notation $[*]$. With TRLWE, we encrypt polynomial values and use the notation $[*]^{(r)}$. The TRGSW encryption scheme also encrypts polynomial values and therefore we use the same notation $[*]^{(r)}$. However, the rare times when we use the TRGSW encryption scheme will be indicated clearly. We hope these choices make our presentation clear and simple.

### 5.3.5 The security of TLWE-based encryption schemes

The security of TLWE-based encryption, as is the case with all LWE-based schemes, is ever-changing. Standards have not yet been set as of writing and new cryptanalysis results have changed and will change the accepted security estimates in the literature. This, in itself, is not cause for concern: the parameters of such schemes need to be kept up to date. This, in turn, will have an impact on their time performance, which will decrease.

Apart from the fact that the TLWE problem is set in the torus ($\mathbb{T}$) a noticeable difference with the original LWE (or RLWE) problem (presented in Section 5.2) is the fact that the secret $\vec{s}$ is taken from a "small" binary set ($\mathbb{B}_N[X]^k$) instead of a "bigger" set such as $\mathbb{Z}_q^n$ or $\mathfrak{R}_q$. In [Bra+13], the authors show that the *small secret* variation of LWE remains hard. However, there are specific attacks that can be used against schemes with small secrets. A recent example is that of [EJK20] - archived in May 2020 - in which the authors describe an attack on small secret LWE that has lead to a reevaluation of TFHE parameters.

Importantly, all of our results are based on previous security estimates for TFHE and therefore do not take into account this reevaluation. Again, even without this work, we would have had to assume that all of the parameters we give for our implementations - and therefore our timings - were subject to change.

### 5.3.6 Torus notations

We present our results in the form of novel homomorphic applications in Part II. To do this, and for clarity sake, we separate our clear-domain solutions for the given problems from our encrypted-domain solutions. The idea for the solution is presented in clear form before the challenges relating to applying such an algorithm in the encrypted domain are presented and tackled.

This means that we move from variables that exist as integers ($\mathbb{N}$) or reals ($\mathbb{T}$) to variables that exist in the torus ($\mathbb{T}$). A given $i \in \mathbb{N}$, an initial input of the clear-domain solution, is therefore rescaled to $t_i = \frac{i}{b}$ with a given *base* $b$. Importantly, the choice of the base determines how the rest of the homomorphic computation plays out:

- *Too large.* If the base is too large, we might loose some precision on the initial value $i$. Indeed, the precision of any implementation is limited and we cannot store torus values with infinite precision.
- *Too small.* If the base is too small, the homomorphic computation might "overflow". For instance, take $i = 1$ encoded as $t_i = \frac{1}{4}$ with base $b = 4$. Then $5 \times i = 5$ but $5 \times t_i = \frac{5}{4} = \frac{1}{4}$. When decoded, after the homomorphic computation is done, we obtain an incorrect result $5 \times i = i$.

The choice of the base can change, the TFHE encryption schemes allows us to do that dynamically in-between operations. This means we can always choose the most appropriate base value. We strive to make it clear when we are talking about values in the torus or integer/real values so that no confusion can be made. The rescaling of a value $i$ in the torus is denoted $t_i$.

### 5.3.7 Torus representation

In all of the thesis, we use a unified representation of torus values that presents the torus as a circle. Figure 5.1 illustrates this through encoding of a real value, its encryption, decryption and

decoding. The figure shows how managing the error and its propagation is crucial to ensuring a correct decryption of the final result.

### 5.3.8 TFHE Operations

In this section we present the different operations in TFHE and their noise propagation. We give them a number that will be used for identification in Table 5.2.

- Internal Addition/Subtraction : $[*] \times [*] \rightarrow [*]$ (1)
  Given two ciphertexts $[\mu_1]$ and $[\mu_2]$, we can add them (resp. subtract them) and obtain a ciphertext $[\mu_1 + \mu_2]$ (resp. $[\mu_1 - \mu_2]$.). The exact same operation can be applied to polynomial ciphertexts.

- Internal Multiplication : $[*]^{(r)} \times [*]^{(r)} \rightarrow [*]^{(r)}$ (2)
  Given an integer polynomial $m[X]$, a TRGSW ciphertext of $m[X]$ $[m]^{(r)}$ a torus polynomial $\mu$ and its TRLWE ciphertext $[\mu]$, we can multiply them and obtain a TRLWE ciphertext $[m \times \mu]$. Note the fact that it takes the use of the TRGSW encryption scheme to obtain the multiplicative homomorphic property.

- External Multiplication : $* \times [*] \rightarrow [*]$ (3)
  Given an integer scalar $m$ and a ciphertext $[\mu]$, we can multiply them and obtain a ciphertext $[m \times \mu]$. The exact same operation can be applied to polynomial ciphertexts with polynomials $\mu[X]$ and $m[X]$. This actually just corresponds to several iterations of internal additions or subtractions.

- Extraction : $[*]^{(r)} \rightarrow [*]$ (4)
  From a ring ciphertext of a polynomial $\mu[X] = \sum_{i=0}^{N-1} \mu_i^i$, it is possible to extract a scalar ciphertext of a single coefficient of $\mu_p$ at a position $p \in \{0, N-1\}$. We can do this at no cost to the noise of the ciphertext. We can extract similarly the corresponding scalar secret key from the initial ring secret key. This means the owner of the initial ring secret key can also decrypt the extracted ciphertext. The extraction operation over a ciphertext $c$ at position $p$ will sometimes be written as $\mathsf{SampleExtract}\,(c, p)$.

- Public Rotation : $[*]^{(r)} \rightarrow [*]^{(r)}$ (5)
  Given a ring ciphertext of a polynomial $\mu[X]$ and an integer $p$ we can obtain a ciphertext of $\mu \times X^p$ with no noise increase.

- Key-Switch : $[*]_s \rightarrow [*]_{s'}$ (6)
  From two keys $s$ and $s'$, we can create an object $\mathsf{KS}_{s \rightarrow s'}$ ($\mathsf{KS}$ for short) called the key-switching key. Without going into details, the key-switching key is an encryption of a decomposition of $s$ using key $s'$. The decomposition is made with a base $\mathsf{base}$ up to precision $t$. Given a ciphertext $[\mu]_{s,\alpha}$ of a scalar value $\mu$ encrypted using the key $s$ with noise $\alpha$, and this key-switching key $\mathsf{KS}$, we can obtain a ciphertext $[\mu]_{s',\alpha'}$ which encrypts the same value under the key $s'$ and with a higher noise $\alpha' > \alpha$.

- Sign Bootstrapping : $[*]_{s,\alpha} \rightarrow [*]_{s',\alpha_b}$ (7)
  From two keys $s$ and $s'$, we can create an object $\mathsf{BK}_{s \rightarrow s'}$ ($\mathsf{BK}$ for short) called the bootstrapping key, with a precision depending on parameters $\ell$ and $B_g$. From these, we define two parameters: $\beta = B_g/2$ and $\epsilon = \frac{1}{2B_g^\ell}$. Given an integer $b$, a ciphertext $[\mu]_{s,\alpha}$ of a scalar value $\mu$ encrypted using the key $s$ with noise $\alpha$, and this bootstrapping key $\mathsf{BK}$, we can obtain a ciphertext $[\mu_0]_{s',\alpha_b}$ where $\mu_0 = 1/b$ if $\mu \in [0, \frac{1}{2}]$ and $\mu_0 = 0$ if $\mu \in [\frac{1}{2}, 1]$. Very

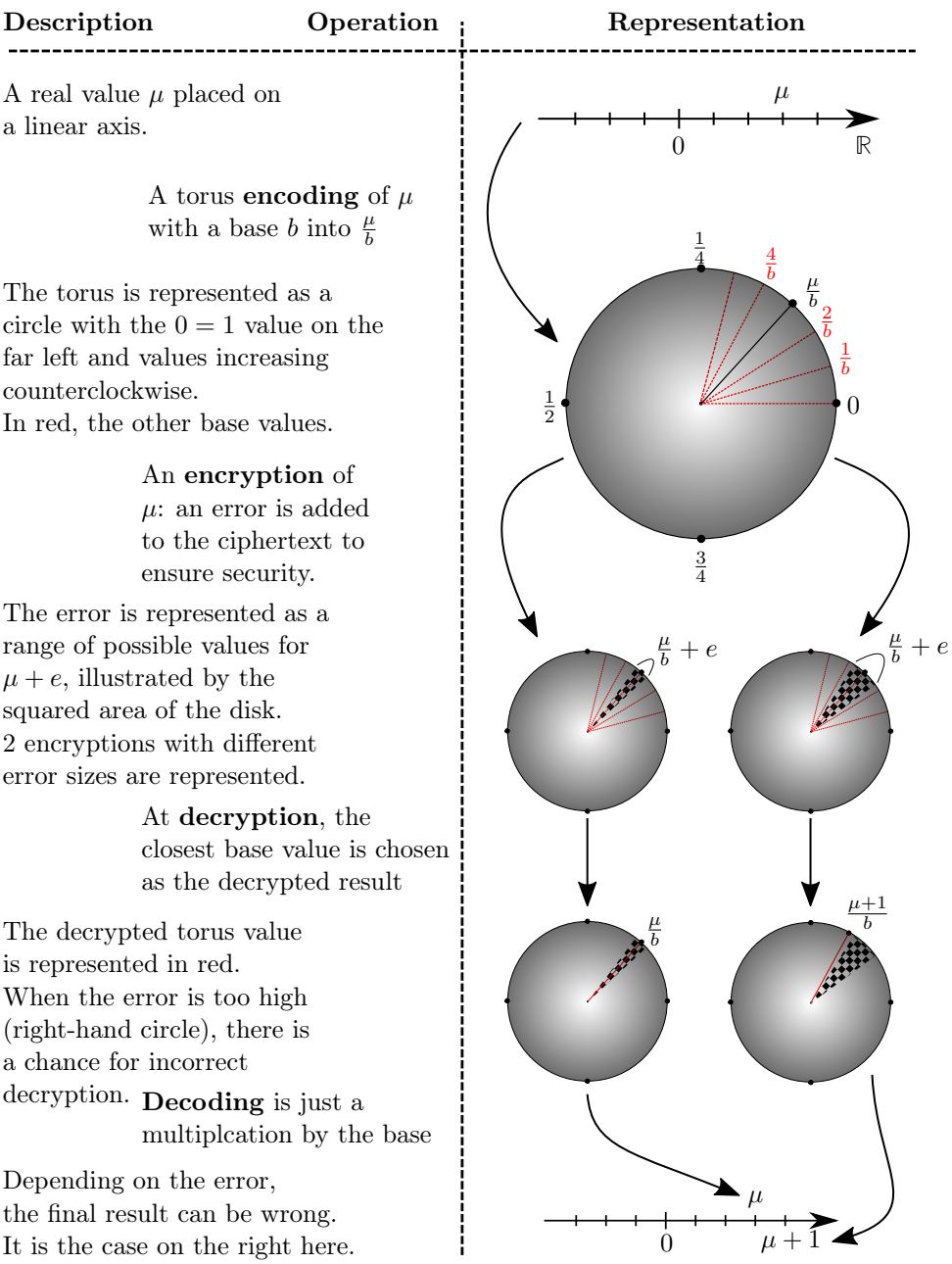


Figure 5.1: A figure presenting our torus representation through encoding, encryption, decryption and decoding.

importantly, $\alpha_b$ is fixed by the parameters of the bootstrapping key $\mathsf{BK}$ and does not depend on the initial standard deviation. We call it *sign bootstrap* because the function that it applies (it could apply other functions) can be considered a sign computation. Indeed, if a value in $[0, \frac{1}{2}]$ is considered positive and a value in $[-\frac{1}{2}, 0]$ (lower half of the torus) is considered negative, the operation outputs 0 for a negative input and $\frac{1}{b}$ for a positive input.
This operation therefore allows us to both apply a sign function to the input ciphertext and reduce its noise down to $\alpha_b$. Figure 5.2 is a representation of this operation. In that figure, we use the same notations as defined in Section 5.3.7. The application of the function is not infinitely precise. The figure illustrates that there is a range of inputs (the red zones) for which the operation does not necessarily output the correct value: it will output a random value. This is not a problem when the parameters are chosen appropriately.
The sign variant of the original bootstrapping algorithm is presented in detail as Algorithm 1.

- Circuit Bootstrapping : $[*] \to [*]^{(r)}$ (8)
  Given a scalar ciphertext $[\mu]$, we can output a $\mathsf{TRGSW}$ ciphertext $[\mu_0]^{(r)}$ where $\mu_0 = 1$ if $\mu \in [0, \frac{1}{2}]$ and $\mu_0 = 0$ if $\mu \in [\frac{1}{2}, 1]$. $\mu_0$ is an integer polynomial of degree 0. This operation is composed of several bootstrap and key-switch operations and outputs the result of a sign bootstrap as seen above, only under a $\mathsf{TRGSW}$ encryption. We exclusively use this operation in order to then apply a TFHE MUX gate.

- TFHE MUX gate: $[*] \times [*]^{(r)} \times [*]^{(r)} \to [*]^{(r)}$ (9)
  Take a $\mathsf{TRGSW}$ ciphertext $[b]^{(r)}$ of message $b \in \{0, 1\}$. $b$ is an integer polynomial of degree 0. Given two TRLWE ciphertexts $[\mu_1]^{(r)}$ and $[\mu_2]^{(r)}$. We can apply a MUX gate that outputs $[b?\mu_1 : \mu_2]^{(r)}$, which is a ciphertext of $\mu_1$ if $b = 1$ and a ciphertext of $\mu_2$ if $b = 0$.

**Remark.** Importantly, our definition of the notions of "Internal Multiplication" and "External Multiplication" *does not match* with the notions defined in [Chi+16a]. They introduce the internal multiplication as one among TRGSW ciphertexts (that we do not use and therefore do not present in this work) and the External Multiplication as *our* Internal Multiplication. The change was made for simplicity sake.

---

**Algorithm 1** The original sign bootstrapping operation. It is a more specific version of the general bootstrapping operation presented in [Chi+16a]. Moreover, we don't end the operation with a key-switching operation as the original paper does. The symbol $\boxdot$ represents the internal multiplication operation.

---

**Input:** A TLWE sample $(a, b) = [\mu]$ of an unknown message $\mu \in \mathbb{T}$, a bootstrapping key $\mathsf{BK}$, a bootstrapping base $b_\delta$.

**Output:** A TLWE sample $u \in \mathsf{TLWE}\left(\frac{1}{b_\delta} \text{ if } \mu \in \left]0, \frac{1}{2}\right[ \; ; \; -\frac{1}{b_\delta} \text{ else}\right)$

1: Let $\overline{b} = \lfloor 2Nb \rceil$ and $\overline{a}_i = \lfloor 2Na_i \rceil$ for each $i \in [1, n]$
2: Let testv $= \left(1 + X + \cdots + X^{N-1}\right) \times X^{-\frac{2N}{4}} \cdot \frac{1}{b_\delta}$
3: $\mathsf{ACC} \leftarrow \left(X^{\overline{b}} \cdot (0, \text{testv})\right)$
4: **for** $i = 1$ **to** $n$
5: $\quad \mathsf{ACC} \leftarrow \left[h + (X^{-\overline{a}_i} - 1) \cdot \mathsf{BK}_i\right] \boxdot \mathsf{ACC}$
6: $u = \mathsf{SampleExtract}\,(\mathsf{ACC}, 0)$

---

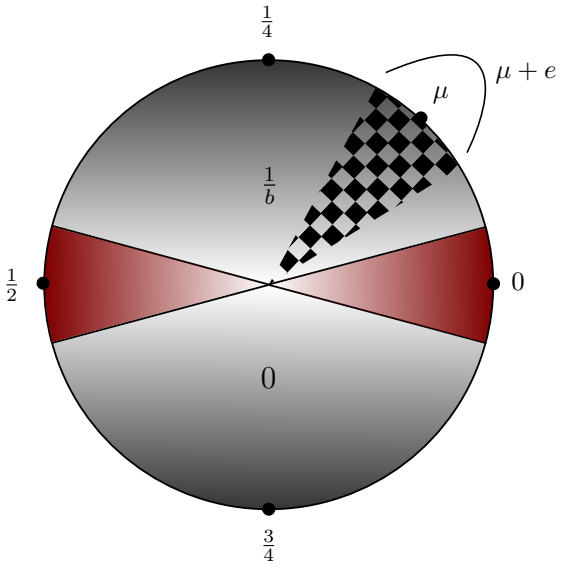


Figure 5.2: The bootstrapping operation represented on the torus. As indicated in the figure, any value in the upper half of the torus will yield an encrypted output of $\frac{1}{b}$ with $b$ a given base; and a value in the lower half an encrypted output of 0. There is a range (red zones) around 0 and around $\frac{1}{2}$ where the bootstrapping operation will return a random value.

## 5.4 B/FV and Chimera

The B/FV (Brakerski / Fan-Vercauteren) scheme is based on the RLWE problem and its encryption algorithm is almost identical to that of the TRLWE encryption algorithm: the main difference is that, instead of $\mathbb{T}[X]/(X^N + 1)$, their message space is $(\mathbb{Z}/p\mathbb{Z})[X]/(X^N + 1)$ with $p$ a given integer called the *plaintext modulus*. A message $\mu$ is then encrypted on the ring $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1)$ with $q$ a higher integer called the *ciphertext modulus*.

We are not going to expand on the B/FV encryption process for the reason that it was unified with that of TFHE under a single framework: Chimera [Bou+20]. That framework encompasses three popular Ring-LWE based homomorphic encryption schemes: B/FV, TFHE and CKKS [Che+17]. Their work is far reaching and contains much more than what we use or present here, and we encourage the interested reader to have a look for themselves.

For our purposes, we only need to know that all B/FV ciphertexts can be transformed into TRLWE ciphertexts with a simple rescaling by $q$, the ciphertext modulus. This simple operation has to be taken into account when considering the standard deviation parameter $\sigma$ for the initial encryption. It will not have the same value depending on the encryption scheme used, and that can lead to confusion. This is where the use of the noise parameter $\alpha$ is convenient. We recall that, in the B/FV setting, $\alpha = \frac{\sqrt{2\pi}\sigma}{q}$ and in the TFHE setting, $\alpha = \sqrt{2\pi}\sigma$. The rescaling by $q$ is therefore inherently present in the B/FV setting, making both definitions of $\alpha$ valid.

The work that we present only uses one homomorphic operation over B/FV ciphertexts: the equivalent to the TFHE external multiplication presented above.

## 5.5 TFHE and B/FV parameters

We implemented all of the schemes that we present in this thesis and, in order to make our results reproducible, we share the parameters we used in every implementation. In Table 5.1, we summarize all of the general parameters used in the TFHE and B/FV encryption schemes. Parameters specific to each problem we address will be presented in their respective chapters.

| Parameter | Description |
|---|---|
| $k$ | a parameter always set to 1 in the general module-LWE setting, so as to be reduced to a ring-LWE setting. |
| $N$ | the size of polynomials in ring-LWE encryption. |
| $n$ | the size of the key in a TLWE encryption. In the TRLWE/TLWE equivalency, we have $n = kN$ |
| $\ell$ | in the original GSW scheme, $\ell$ is fixed at $\ell = \lceil \log q \rceil$ for the GSW encryption to be exact. In the TRGSW encryption, $\ell$ is a stand-alone parameter that, along with $B_g$, defines the precision of the encryption. |
| $B_g$ | the base for the approximate gadget decomposition in the TRGSW encryption scheme. |
| $\epsilon$ | a variable linked to $\ell$ and $B_g$ and used only for simplicity of notations purposes: $\epsilon = \frac{1}{2B_g^\ell}$ |
| $\beta$ | a variable directly linked to $B_g$ and used only for simplicity of notations purposes: $\beta = B_g/2$ |
| $t$ | the precision of the decomposition of the encrypted key in the creation process of the key-switching key |
| base | the base for the decomposition of the encrypted key in the creation process of the key-switching key |
| $\sigma$ | the standard deviation of the Gaussian distribution used for encryption. |
| $\alpha$ | the noise parameter is used for a noise equivalence between TFHE and B/FV. The security of the scheme directly depends on $\alpha$ and $N$. |
| $p$ | the modulus of B/FV plaintexts. |
| $q$ | the modulus of B/FV ciphertexts. |
| $r_p(q)$ | the remainder of the division of $q$ by $p$: $q = \lfloor q/p \rfloor + r_p(q)$. It is an important factor in the noise propagation equations of B/FV operations. |

Table 5.1: TFHE and B/FV parameters used to implement both schemes in our work.

## 5.6 Noise propagation

We have mentioned several times that almost all operations in TFHE or B/FV have a cost: the ciphertext noise increases as we go, until decryption is no longer accurate or until a bootstrapping operation resets the noise. The bootstrapping operation is quite a bit heavier computationally

than many of the simpler ones, and this means we want to apply this noise reset only when we absolutely need to. In order to do this, we present in Table 5.2 the average noise outputs for all of the homomorphic operations presented above. With these equations, one can run simulations over clear data to determine the optimal structure for their homomorphic algorithm (which is what we do to determine our optimal parameters in our work). In the table, TFHE operations are distinguished using their respective number as introduced in Section 5.3.8 and the last operation is the B/FV external multiplication operation.

| # | Operation | Noise |
|---|---|---|
| (1) | $[\mu_1] + [\mu_2] \to [\mu_1 + \mu_2]$ | $\vartheta_1 + \vartheta_2$ |
| (2) | $[m]^{(r)} \times [\mu_2]^{(r)} \to [m \times \mu_2]^{(r)}$ | $A\vartheta_m + \|m\|^2 (B + \vartheta_2)$ |
| (3) | $m \times [\mu_2] \to [m \times \mu_2]$ | $\|m\| \cdot \vartheta_2$ |
| (6) | $[\mu_1]_s \to [\mu_1]_{s'}$ | $\vartheta_1 + nt\vartheta_{\mathsf{KS}} + n\mathsf{base}^{-2(t+1)}$ |
| (7) | $[\mu_1]_{s,\alpha} \to [\mu_1]_{s',\alpha_b}$ | $\vartheta_{\text{boot}}$ |
| (8) | $[\mu_1] \to [b]^{(r)}$ | $\vartheta_{\text{boot}} + nt\vartheta_{\mathsf{KS}} + n\mathsf{base}^{-2(t+1)}$ |
| (9) | $[b]^{(r)} ? [\mu_1]^{(r)} : [\mu_2]^{(r)} \to [b ? \mu_1 : \mu_2]^{(r)}$ | $\max(\vartheta_1, \vartheta_2) + A\vartheta_b + B$ |
| B/FV | $m_1 \times [m_2] \to [m_1 \times m_2]$ | $N \|*\| m_1 (\|*\| e_2 + r_p(q)/2)$ |

Table 5.2: Elementary operations for TFHE and B/FV and their noise overhead In this table, we use $m$, $m_1$ and $m_2$ as integer polynomials, $b$ as an integer value of either 0 or 1, and $\mu_1$ and $\mu_2$ as torus scalars or polynomials (depending on the context). $\vartheta_i$ represents the variance of ciphertext $[\mu_i]$ before the operation. In the B/FV case, we use $e_2$ as the *inherent noise* (see [CLP17]) of ciphertext $[m_2]$. The same goes for $\vartheta_m$ and $m$. We present only the operations for which there is a change in the output noise of the ciphertext. This means that the Extraction (4) and Rotation (5) operations are not included. $\vartheta_{\mathsf{KS}}$ is the variance used for the encryption of the key-switching key $\mathsf{KS}$ and $\vartheta_{\mathsf{BK}}$ for the bootstrapping key $\mathsf{BK}$. In the case of the bootstrapping, the output variance overhead is a constant given by: $\vartheta_{\text{boot}} = 2Nn(k+1)\ell\beta^2 \times \vartheta_{\mathsf{BK}} + n(1+kN)\epsilon^2$. $A$ and $B$ are constants with values $A = (k+1)\ell N\beta^2$ and $B = (kN+1)\epsilon^2$.

# Part II

# Contributions

# Chapter 6

# An homomorphic Hopfield network

**Contents**



## 6.1 Introduction

The work that we present in this chapter was made to answer two questions. Are there machine leaning algorithms that naturally integrate efficiently to FHE ? Can we provide a "full-encryption" property where both the model and the input to be classified would be encrypted ?

Our interest in the first question was in finding alternative applications for existing FHE tools. Indeed, making FHE applications practical in real-world scenarios can come both from works that improve on existing FHE schemes and from works that find new efficient ways to use existing tools. Eventually, we settled on using Hopfield networks, a type of recursive neural network. Using a Hopfield network, we designed, parameterized and implemented two confidentiality-preserving face-recognition algorithms. In one of them, an encrypted query is sent through a clear Hopfield network. In the second, an encrypted query is sent through an encrypted Hopfield

network. In particular, this leads us to efficiently evaluate $2^{\text{nd}}$ degree polynomial functions homomorphically.

We published our findings in [ISZ19], and this chapter is based on that article.

## 6.2 Context and Motivations

### 6.2.1 Hopfield Networks Basics

Hopfield networks are recursive neural networks with only one layer of neurons, all connected. They were first introduced by J. J. Hopfield in 1982 in [Hop82; SZ07]. See also [Mac03] and [Gur97] for further uses in the literature.

Let us thus consider a Hopfield network made up of $M$ neurons. Every neuron is numbered from 1 to $M$ and given an *activation value*. Neuron $i$ has an activation value $a_i$. In most neural networks, the activation value is a real number. Here, because we are considering a *discrete Hopfield Network*, we consider only binary activation values: $a_i \in \mathbb{B}$. Every neuron is connected to the other neurons and their connection is given a weight: $w_{i,j}$ is the weight of the connection between neurons $i$ and $j$.

Training a Hopfield network - as is the case for other neural networks - means finding appropriate values for the weights $w_{i,j}$. The evaluation phase requires a binary query vector of the size of the network ($M$) and the activation values of each neuron take the value of their respective coefficient in the query vector. Then, the network updates these values at every iteration according to a correlation property. The weight $w_{i,j}$ between two neurons $i$ and $j$ represents the correlation that exists between them. An update of neuron $i$ translates to a weighted sum of all other neuron values with an *activation function* $\theta$:

$$a_i = \theta\left(\sum_{j=1}^{n} a_j w_{i,j}\right)$$

This update can happen neuron by neuron randomly or in order. The update can also happen simultaneously for all neurons. Experimentally, we found that a sequential, fixed-order update gave the best results in our case. In our case, the activation function is a sign function:

$$\begin{aligned} \theta(x) &= 1 \quad \text{if} \quad x \geq 0 \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Essentially, every neuron is "drawn" towards the values of the neurons most correlated to it. This correlation is symmetric ($w_{i,j} = w_{j,i}$) and a neuron is not correlated with itself ($w_{i,i} = 0$). In practice what the Hopfield network does naturally is store a number of patterns by highly correlating the bits of the patterns. The network does not need to globally converge to be useful. If we define certain bits as "classification bits" (for example, in our face recognition experiment only 3 bits out of 256 are used to classify between up to 8 patterns), then only those bits need to be updated to obtain a first result. With a sign function as an activation function these networks are computationally lighter than their feed-forward counterparts.

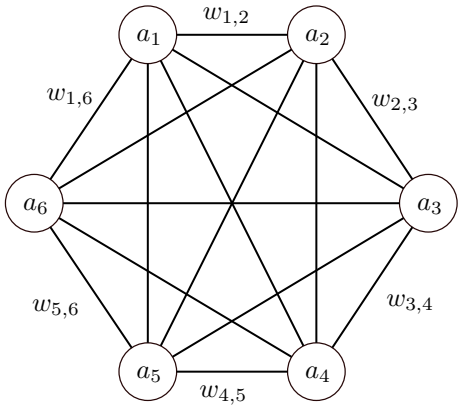


Figure 6.1: A toy Hopfield Network example, with only 6 interconnected neurons. The inner weights are not represented for clarity sake.

### 6.2.2 Activation functions in FHE applications

A quick remark here to give the reader some more context on the choice of activation function that we made. The "activation function" we speak of is the same as can be found in other types of neural networks and therefore usually corresponds to a non-linear function such as a sigmoid ($\theta(x) = (1 + e^{-x})^{-1}$) or a rectified linear function ($\theta(x) = \max(0, x)$) also known as ReLU. Homomorphic machine learning algorithms have a hard time approximating those functions and therefore choose to either replace them with a polynomial [Dow+16] or a sign function [Bou+18]. In the case of a sign function, the output is binary. Hence, the activation values have to be binary, therefore reducing somewhat the accuracy of the network. Sometimes, this loss can be necessary because of the great noise cost polynomial functions bring in an homomorphic algorithm. We followed the example of [Bou+18] in choosing to use the sign for the main reason that it allows us to build a fully homomorphic classification scheme by applying the sign function through the bootstrapping operation. The choice of a polynomial function would have meant a levelled scheme.

### 6.2.3 Scenario

The partially-encrypted case corresponds to one that we have touched on in Chapter 3 (Section 3.4.3). The general use-case presented in Figure 3.2 is applicable: a Hopfield network is hosted by its owner, and a user sends an encrypted query for classification. The encrypted result is sent back for decryption.

The interest in adding a layer of encryption on the network lies in the possibility that the owner of the network and the computation handler are not the same entity and that both entities do not trust each other. Importantly, both the user and the network owner have to use the same encryption key for the homomorphic operations to work. One could imagine a number of ways to make this work, depending on the specific constraints in a real-world scenario (time, bandwidth and storage capacity, trust,...). Here, we present one possible solution, where both the querier and the network owner encrypt their data under the querier's public key and send it to a *computation host* that makes the homomorphic classification and returns the result to the querier. The computation host then needs to be trusted to not collude with the querier. That

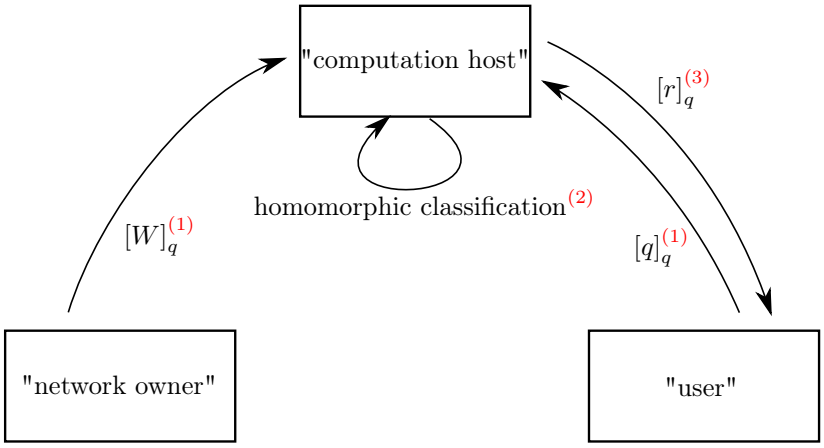


Figure 6.2: A figure illustrating a possible protocol for secure classification over a remote server, in the fully-encrypted case. A query $q$ is sent encrypted by the querier to the computation host, and the network weight matrix $W$ is sent by the network owner to the computation host under the querier's public key. The result $r$ is sent back to the querier for decryption. the notation $[*]_q$ is used to denote that data is encrypted using the querier's public key. In red, the step number in the protocol.

protocol is illustrated in figure 6.2.

## 6.3 Homomorphic Hopfield Network design

In this work, we present two alternatives for confidentiality-preserving Hopfield Network classification. In Section 6.3.1, we present the case where the network weights are not encrypted, but the input activation vector is. The scheme remains essentially the same when the network weights are encrypted instead, and the input activation vector is not. We call that scheme the *partially encrypted* scheme. In Section 6.3.2, both the network weights and the activation vector are encrypted. We call that scheme the *fully encrypted* scheme.

### 6.3.1 Encrypted inputs through a clear network

In this section, we present the partially-encrypted case.

We start by encrypting the activation values $a_i$ as TLWE ciphertexts $c_i = \left\lceil \frac{1+2a_i}{2S} \right\rceil$. Now, in order to update the activation value of one neuron, we need to compute a scalar product and take the sign of the result and then insert it back into the network for further computations. For a given neuron $p$, this corresponds to the following computation: $a_p = \mathsf{sign}\left(\sum_{i=1}^{M} a_i w_{p,i}\right)$. The TLWE ciphertexts $c_i$ are grouped in a vector $\vec{c}$ and the weights $w_{i,p}$ are clear. Assuming we want to update the $p^{\text{th}}$ neuron, we have a simple, fully homomorphic solution presented in Figure 6.3. In it, we see that the bootstrapping operation is used to apply a sign function, as well as reset the noise. Because the bootstrapping operation naturally changes the key of the input ciphertext, a key-switching has to be applied before we can return the updated activation value. This operation is then repeated on a given number of activation values to update them

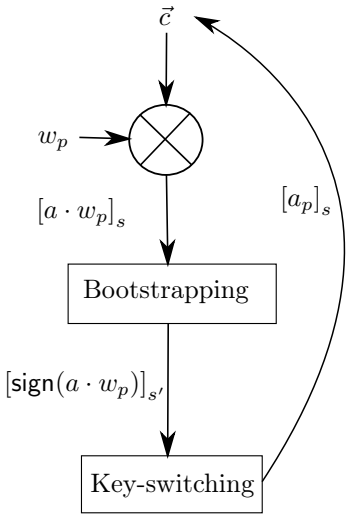


Figure 6.3: This figure illustrates the partially-encrypted case update algorithm. The output of the key-switching operation is reincorporated into the vector of ciphertexts as its $p^{\text{th}}$ component. The initial encryption key is denoted $s$ here and the intermediate key (between the bootstrapping and the key-switching) is denoted $s'$.

homomorphically.

If we choose the parameters of the scheme and $\vartheta_c$ such that

$$MB^2 \left(\vartheta_{\text{boot}} + nt\vartheta_{\mathsf{KS}} + n\mathsf{base}^{-2(t+1)}\right) \leq \vartheta_{\max} \quad \text{and} \quad \vartheta_c = \frac{\vartheta_{\max}}{MB^2}$$

then we ensure a correct final decryption, whatever the number of iterations. See Sections 5.5 and 5.6 for explanations on the parameters used above and their interaction in the noise propagation equations of TFHE operations.

### 6.3.2 Encrypted inputs through an encrypted network

We now consider the case where both the weights and the activation values are encrypted. We have $\vec{a} = (a_0, \ldots, a_{M-1})$ and $\vec{w_p} = (w_{0,p}, \ldots, w_{M-1,p})$ for every $p$.

**Encryption Methodology.** We use the TRLWE and TRGSW encryption schemes to encrypt the activation values and the weights respectively, with the same key.

We constrain $N \geq M$. We first encode the data as two $(N-1)$-degree polynomials (with 0 coefficients to pad if need be):

$$\forall p \in [0, M-1], \quad W_p = \sum_{i=0}^{N-1} w_{i,p} \cdot X^{N-1-i}, \quad A = \sum_{i=0}^{N-1} a_i \cdot X^i \in \mathbb{T}_N[X]$$

We then encrypt both the weight polynomials and the activation polynomial respectively as TRGSW and TRLWE ciphertexts: $[W_p]^{(r)}$ and $[A]^{(r)}$. Both are encrypted using the same noise parameter $\alpha_c$.

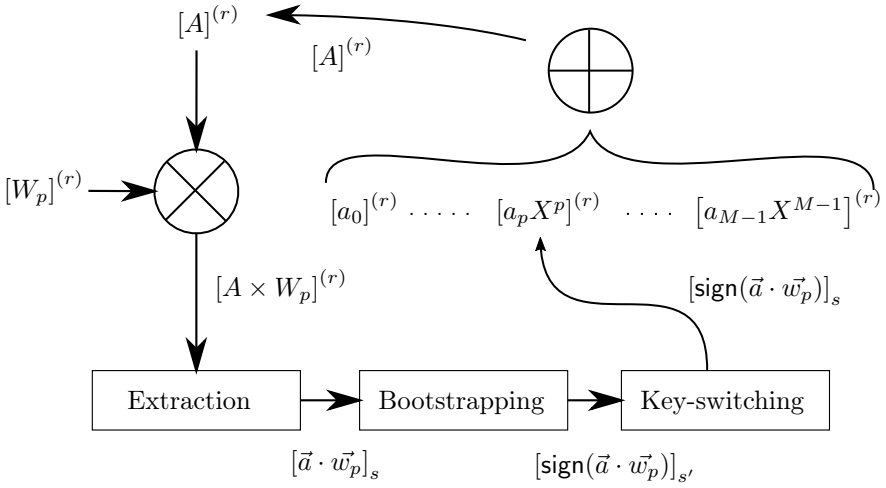


Figure 6.4: This figure illustrates the update of the $p^{\text{th}}$ neuron in the fully-encrypted case. There is first an internal product that produces the scalar product we are looking for as the last coefficient of the output polynomial. After extraction of the value, sign computation through the bootstrapping operation, and key-switching, the original activation ciphertext $[A]^{(r)}$ is recreated by summing all of the $C_i$ ciphertexts once $C_p$ has been updated. The initial encryption key is denoted $s$ here and the intermediate key (between the bootstrapping and the key-switching) is denoted $s'$.

We also encrypt $M$ TRLWE ciphertexts (with the same noise parameter again) for each individual activation value:

$$\forall i \in [0, M-1], \quad C_i = \left[a_i X^i\right]^{(r)}$$

We group them in a vector of ciphertexts $\vec{C} = (C_0, \ldots, C_{M-1})$.

**Update Algorithm.** Figure 6.4 presents an update of the $p^{\text{th}}$ activation value for a given $p$. Note that the key-switching operation hides a rotation operation that transforms $[a_p]^{(r)}$ into $[a_pX^p]^{(r)}$ at no cost to the error propagation and virtually no time cost. Keeping TRLWE ciphertexts of individual activation values (the $C_i$ ciphertexts) allows us to rebuild a new and updated $[C_a]^{(r)}$ ciphertext by summing them all at the end.

**Correctness.** There are two sets of parameters defined both for the initial ciphertexts and for the bootstrapping key $\mathsf{BK}$. We will therefore use the notation $*_c$ for the initial ciphertext parameters and $*_b$ for the bootstrapping key parameters. As for the key-switching key $\mathsf{KS}$, we will use the notation $*_s$. We have $N_s = N_c$. This can work because all of the input ciphertexts have the same parameters with only one exception: the variance of the $C_p$ ciphertext which is not equal to $\vartheta_c$; we will refer to it by $\vartheta'_c$. For this next theorem, we set some constants in order to simplify notations. We set $\mathbf{A} = 4N_cN_bl_b\beta_b^2$ ; $\mathbf{B} = N_c(1+N_b)\epsilon_b^2$ ; $\mathbf{C} = N_ct$ ; $\mathbf{D} = N_c\mathsf{base}^{-2(t+1)}$ ; $\mathbf{E} = MB^2$ ; $\mathbf{F} = \epsilon_c^2(1+N_c)$ and $\mathbf{G} = 2\ell_cN_c\beta_i^2$.

Then, if we choose the parameters of the scheme and $\vartheta_c$, $\vartheta'_c$ such that

$$M(\mathbf{G}+\mathbf{E}) \times (\mathbf{A}\vartheta_{\mathsf{BK}} + \mathbf{B} + \mathbf{C}\vartheta_{\mathsf{KS}} + \mathbf{D}) + \mathbf{F} \times \mathbf{E} \leq \vartheta_{\max}$$

$$\vartheta_c \leq \frac{\vartheta_{\max} - \mathbf{F} \times \mathbf{E}}{\mathbf{G}+\mathbf{E}} \quad \text{and} \quad \vartheta'_c = \mathbf{A}\vartheta_{\mathsf{BK}} + \mathbf{B} + \mathbf{C}\vartheta_{\mathsf{KS}} + \mathbf{D}$$

then our computation is fully homomorphic.

## 6.4 Practical Implementation

### 6.4.1 Face recognition via Hopfield networks

Although their applicability is less universal than feed-forward neural networks, Hopfield networks are known to perform well for certain tasks including image analysis and recognition. In this section, we provide experimental results in the case of a face recognition function. We used a database[1] consisting of pictures of 153 individuals with 20 pictures per individual with different facial expressions on each person. The feature extraction for the images was done using the LTP (Local Ternary Patterns) introduced in [TT10] and from the Java library Openimaj [HSD11]. From a face image, the feature extraction gives us 128 $80 \times 80$ matrices of raw binary data that we use for the classification. For this test we selected 8 random faces. By training the network on those patterns we wish to be able to classify a picture of a person as one of those 8 faces. For this we apply one iteration of classification on 3 neurons of a 256-neuron network. Empirically, the network stabilizes after the first iteration, hence we do not find better classification results with more iterations. We find that reducing the number of neurons to 256 maximizes the time/performance ratio, giving us an 86.2% classification performance (however we are confident that better results could be obtained with deeper machine learning know-how).

In terms of performance, we fully implemented our classification and ran tests on an Intel Core i7-6600U CPU. Overall, a classification does not take more than 0.21 seconds in the partially-encrypted case and under 0.6 seconds for the fully-encrypted case.

### 6.4.2 Parameter choices

Choosing the parameters, we have to take into account the security constraints from the underlying LWE problem on top of the correctness constraints for the two schemes we present. The overall security of the scheme relies entirely on the security of the underlying TFHE scheme and therefore the LWE problem on which it is built. For a given size of the key and a given security parameter, we can obtain a minimum variance using the `lwe-estimator` script. The structural parameters of the network are $M = 256$ and $B = 8$. We have $S = M = 256$.

**Parameters for the partially-encrypted case.** For both an 80-bit security and a 110-bit security, the parameters in table 6.1 are appropriate. However we need to choose different values for $n$. We set $n = 470$ for an 80-bit security; And $n = 650$ for a 110-bit security.

[1] https://cswww.essex.ac.uk/mv/allfaces/faces94.html

| $N$ | $\alpha_b$ | $\alpha_c$ | $\mathrm{Bg}_b$ | $\ell_b$ | $t$ | base |
|---|---|---|---|---|---|---|
| 1024 | 4.26e-13 | 2.41e-06 | 64 | 4 | 3 | 32 |

Table 6.1: Parameter values for both security settings in the partially-encrypted case. We have $\alpha_s = \alpha_c$ and $k$ is set to 1.

**Parameters for the fully-encrypted case.** Table 6.2 presents the parameters used for the initial ciphertext encryption, the bootstrapping key and the key-switching key. These parameters are valid for an 80-bit security. We were not able to go beyond this security level: the parameter constraints did not hold anymore.

| $\lambda$ | $\alpha_c$ | $N_c$ | $\mathrm{Bg}_c$ | $\ell_c$ |
|---|---|---|---|---|
| 80 | 4.26e-13 | 1024 | 64 | 4 |

| $\lambda$ | $\alpha_b$ | $N_b$ | $\mathrm{Bg}_b$ | $\ell_b$ |
|---|---|---|---|---|
| 80 | 8.88e-16 | 2048 | 256 | 5 |

| $\lambda$ | $\alpha_s$ | $N_s$ | $t$ | base |
|---|---|---|---|---|
| 80 | 4.26e-13 | 1024 | 4 | 128 |

Table 6.2: Parameter values for initial ciphertext encryption (top left), for the bootstrapping key BK (top right) and for the key-switching key KS (bottom) for an 80-bit security

We provide the sizes of the ciphertexts associated with these parameters. The size of a single initial TRLWE ciphertext is 16.5 KBytes and the size of the initial TRGSW ciphertext 132 KBytes. The size of the Bootstrapping key is 337 MBytes and the size of the Key-switching key 135 MBytes. The total size of the encrypted network is 33.7 MBytes and the total size of the encrypted activation values 4.22 MBytes.

## 6.5 Conclusion

This work was our first result in considering other ways that homomorphic encryption could be applied to machine learning problems. Although the subsequent work we produced and present in this thesis are not built directly upon this one, it is with the same spirit that we endeavored to keep looking for better ways yet make secure, outsourced, machine learning a reality.

A more direct link to our second endeavor (described in this next chapter) exists in the interest one can find in training a Hopfield network over encrypted data. In order to do this, one needs to be able to compute an argmax operation of some kind. This is true for most machine learning algorithms because the selection of the model is made with regard to the highest classification rate. This is illustrated in Figure 3.1. It comes naturally then that second work that we present here is about computing an argmin function homomorphically (an argmax as well, with a minuscule tweak of the algorithm).

# Chapter 7

# An homomorphic argmin operator

**Contents**



## 7.1 Introduction

We mentioned in the last chapter our initial motivations for finding an homomorphic argmax operator. The original idea was for it to be used in the context of an homomorphic training algorithm. This is not the direction that we took immediately. We mention that it is not for lack of results or for a belief that it would not work. In fact, that very line of research was eventually explored in [Séb+20]. We present that work in Chapter 9.

In the end, we first chose to view this argmax tool as a classification means, sticking to our initial line of research which consisted of investigating *inference phase* homomorphic tools. Indeed, although the argmax operator can be quite useful as a training tool, the argmin operator is, in itself, an classification tool, as part of the 1−Nearest Neighbor (1-NN) machine learning algorithm. The 1-NN algorithm consists of matching the class of the query vector to the class of

the closest vector in a pool of model vectors. We present this algorithm in much more detail in Chapter 8, as a subset of the larger set of $k$-NN classifiers.

Indeed, for this work, and to apply our tool to state-of-the-art machine learning, we decided to show-case our results over an embedding-based deep neural network. In that context, we achieve a state-of-the-art, model-confidential speaker recognition scheme in almost real time. Our scheme works with any such embedding-based algorithms: we provide a generic framework for efficient model confidentiality in a set of state-of-the-art speaker/face recognition algorithms.

The work that we present here was published as [ZCS20], and this chapter is based on that article.

## 7.2 Context and Motivations

### 7.2.1 A Neural Embedding system: VGGVox

VGGVox refers to a neural embedding system presented in [CNZ18b]. It was trained and tested on datasets called VoxCeleb1 [NCZ17] and VoxCeleb2 [CNZ18a]. We refer to the cited papers for an in-depth presentation of the system.

**Speaker recognition.** VGGVox is a speaker recognition system. What it is meant to do is store a number of patterns associated with speakers. When a new audio sample is ran through the system, VGGVox associates it with one of the existing ones. Specifically, the system applies two consecutive Convolutional Neural Networks (CNNs) to a raw audio segment containing a speech sample from one given speaker. The second of these networks outputs a real vector embedding of dimension 1024 that is a representation of the speaker in the Euclidean space $\mathbb{R}^{1024}$. The point of the system is to make it so that the Euclidean distance in the output space ($\mathbb{R}^{1024}$) becomes a good measure of the similarity between the speakers of the samples in the input space (raw audio samples). In short, if we find that two audio samples going through the networks yield two vectors that are close-enough, then we can assume that the same speaker has provided the two original samples. The way that we use this system is to have a certain number of reference embeddings each representing a different speaker. Then, we classify a new sample as being spoken by one of the reference speakers by finding the reference embedding that is closest to the new embedding. Therefore, we need to solve a nearest neighbor problem (1-NN).

**VGGVox training and model creation.** With VGGVox, the training of the CNNs and the creation of the model are two different things. The neural networks are not trained for speaker-recognition per se, they are trained to transform speaker similarity in the input space into closeness in the output space. Therefore, the networks are trained once and for all for all possible usage of the system. At this point, the reference speakers the user of the system wants to store come into play for the model creation. For every one of them, a speech sample is ran through the CNNs and the output *model vectors* are stored. In Figure 7.1, which represents both the model creation and the classification phase, these model vectors correspond to the small red stars.

**Scope and performance.** Of course, this kind of embedding-based neural architecture is not universally applicable. However, it has shown to be a very efficient way to solve specific problems. One such system is used in [SKP15] for face recognition. Embedding-based systems are also typical for text-processing applications [GBC16]. Also, [SKP15] and [CNZ18b] show

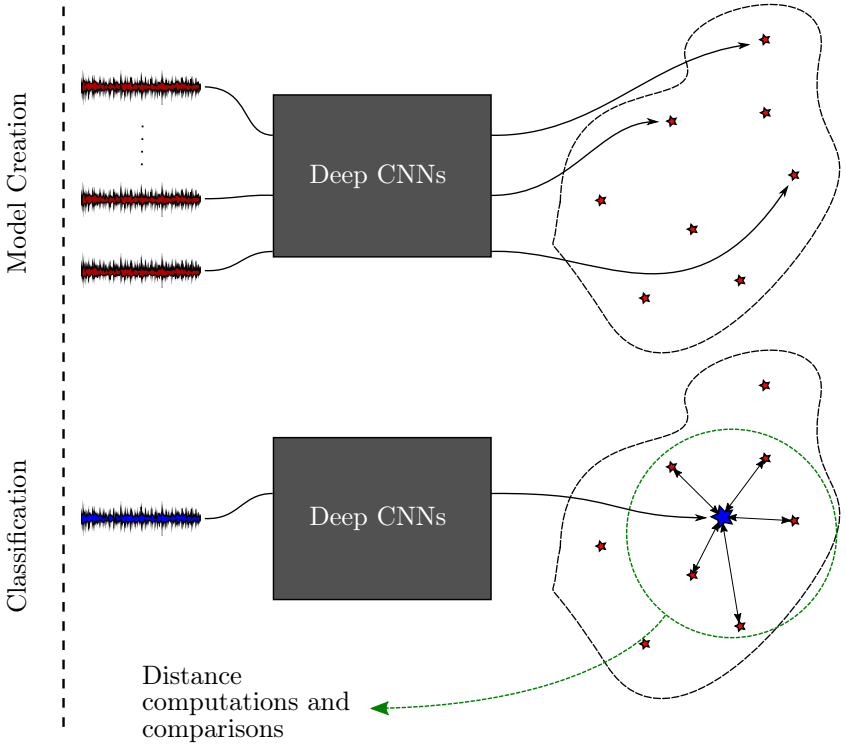


Figure 7.1: A figure that represents both the model creation phase and the classification phase of the VGGVox system. In red, raw audio samples from different speakers are first put through the CNNs for storage as model embeddings (or model vectors). Then, the blue audio sample is classified as coming from one of those model speakers by running it through the CNNs and then finding its corresponding embedding's closest model embedding neighbor. This requires distance calculations and comparisons.

that the field of embedding-based neural systems is evolving quickly and we can expect that other applications will be proposed in the years to come. This is important because the tools that we provide in this work can be generalized to any system working in the same way as the VGGVox system. As long as the measure of similarity that interests us at the output of the neural networks is the Euclidean distance, then the framework we present works without having to be adapted to the specifics of the new system.

We refer to [CNZ18b] for precise information on the performance of VGGVox in terms of classification rate. As such it is recognized as a highly accurate speaker-recognition system that achieves best-in-class performance, in the speech processing community.

### 7.2.2 Scenario

The philosophy underlying the architecture of the VGGVox system, which is similar to that of other embedding-based systems such as Google's FaceNet, is to avoid the burden of retraining the whole system when a new individual needs to be recognized. This is desirable for obvious maintainability reasons. Hence, the overall classification systems is partitioned in a first generic pre-processing step which is trained once and for all on a representative (possibly public) learning

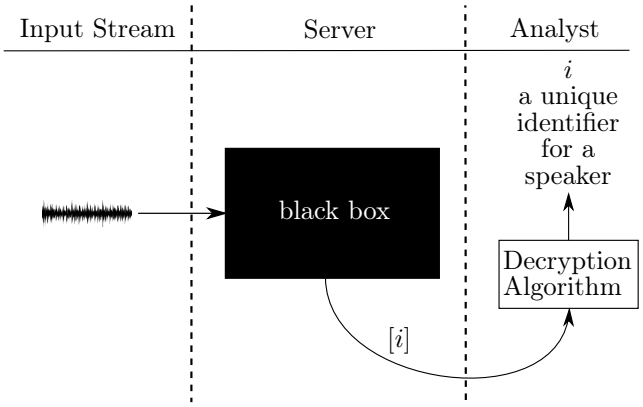


Figure 7.2: A figure illustrating the case study that we describe.

data set, without depending on the individuals that will in fine have to be recognized by the system (which usually are not known at design time). Then, in order to achieve the desired classification function, the pre-processing part is supplemented by a much simpler classification scheme, usually based on some distance, which this time depends on the individuals to be classified. In use cases where one wishes more to hide what is sought in data rather than hiding the data themselves (from some server), marrying FHE with such embedding-based systems is therefore quite relevant since the complex pre-processing part does not need to be executed in the encrypted-domain. This is relevant in use-cases where an entity, the *analyst*, wishes to deploy an analysis "black-box" on an honest-but-curious *server* whose job is to analyze raw traffic (available in non-encrypted form to the server) with respect to an undisclosed analysis criteria. Examples of this include hiding individuals or profiles of interest when analyzing a video stream, hiding cyber-attack signatures when analyzing packet streams or, as illustrated in depth here, identifying undisclosed individuals in voice streams (to name a few examples). In our case, a recording of an individual is accessible to the server, but its identity remains secret. The general architecture is illustrated on Figure 7.2. Furthermore, architectures of this type also have the property that the server has access to the raw data but no access to the analysis criteria or results whereas the analyst has access to the analysis results but no access to the raw data. Such properties are desirable in cases where legal, ethical or commercial reasons prevent a single party to have access to both raw data and analysis results.

To be more concrete with the use of VGGVox in such a system we can consider the following setup where an agency (the analyst) need to perform some analysis of the voice calls carried over the network of an operator (the server). By law, the agency is not allowed to have access to the raw voice calls but may have the right to know whether individuals from a target set are involved in some phone conversations. By law, the operator cannot be disclosed information on the target individuals. In this setup, the analyst, which is owning the FHE public and private key, first runs voice samples from the target individuals through its own copy of the VGGVox network in order to obtain the associated embeddings, say $r_1$, ..., $r_N$, which it encrypts and sends to the operator. On the other side, the operator runs a given voice samples[1] through its own copy of the VGGVox system and obtains an embedding $r$ which it confronts to the encrypted $r_i$'s to obtain (in the simplest case) encryptions of e.g., $\min_i \|r - r_i\|^2$ as well as $\mathsf{argmin}_i \|r - r_i\|^2$ which

[1]The way these samples are selected may depend on some meta-data and is beyond the scope of this work.

it sends to the analyst for decryption (and beyond).

## 7.3 Homomorphic Speaker Recognition

In order to determine homomorphically which encrypted vector from a number of reference embeddings is closest to a plain vector input, first, we need to be able to compute distances, and second, we need to be able to compare those distances. We present in this section the main algorithms for the distance computation and the comparison phase. The initial distance is computed using B/FV ciphertexts and B/FV operations. The distance ciphertexts then become TFHE ciphertexts (again, with a simple rescaling and therefore at no significant cost) for the comparison phase. We will first present the distance computation operations and then two schemes for the comparison of all the distances, each with their strengths and weaknesses.

### 7.3.1 Distance Computation

We are given $d$ real vectors: $c^{(k)} \in \mathbb{R}^\gamma, k \in \{1, ..., d\}$ of dimension $\gamma$. These are the outputs of the VGGVox system for $d$ different speakers. To be more precise, we are going to use an approximation of the actual VGGVox output vectors. We are going to consider $c^{(k)} \in \mathbb{N}^\gamma$ and we refer to Section 7.4.1 for a discussion on the impact of this. We encode each of these vectors in polynomial form:

$$\forall k \in [1, d], \quad C^{(k)} = \sum_{i=0}^{\gamma-1} c_{i+1}^{(k)} \cdot X^i$$

Actually, the entity performing the distance computation, as mentioned in Section 7.2.1, only has an encrypted version of those vectors: $\left[C^{(k)}\right]^{(r)}$. These ciphertexts are the reference ciphertexts and they are stored on the server waiting for a new (cleartext) vector to come in for comparison. Let's call $m \in \mathbb{R}^\gamma$ this new vector and encode it as a polynomial:

$$M = \sum_{i=0}^{\gamma-1} m_{\gamma-i} \cdot X^i$$

Given this $M$ and the $\left[C^{(k)}\right]^{(r)}$, we want to compute $\mathbb{d}_k$ the $d$ distances from $m$ to every one of the $c^{(k)}$. This is a basic computation and we only go into detail so that to assess the multiplicative depth of the computation. We define:

$$\mu = \sum_{i=1}^{\gamma} m_i^2 \quad \text{and} \quad \forall k \in \{1, ..., d\}, \quad \mu^{(k)} = \sum_{i=1}^{\gamma} \left(c_i^{(k)}\right)^2$$

For a given $k$, let's call $\mathbb{D}_k$:

$$[\mathbb{D}_k]^{(r)} = -2 \times M \times \left[C^{(k)}\right]^{(r)} + \mu \cdot X^{\gamma-1} + \left[\mu^{(k)} \cdot X^{\gamma-1}\right]^{(r)} \tag{7.1}$$

Remark that we can pre-compute $-2 \times \left[C^{(k)}\right]^{(r)}$ and $\left[\mu^{(k)} \cdot X^{p-1}\right]^{(r)}$. Then we can obtain $[\mathbb{d}_k]$ by extracting the $(\gamma-1)^{\text{th}}$ coefficient from $[\mathbb{D}_k]^{(r)}$. That is an encryption of the distance between $m$ and the reference vector $c^{(k)}$.

It is straightforward to see that this yields the correct result. What is important here is that, given the pre-computations, the distance only requires one external multiplication, one external addition and one internal addition.

**Alternative method:** As we will see in Section 7.3.3, depending on the method used to compute the argmin, we do not necessarily need to compute the distances themselves but rather all the distance differences $(\mathbb{d}_k - \mathbb{d}_l)$. This means we can do:

$$[\mathbb{D}_k - \mathbb{D}_l]^{(r)} = M \cdot \left[2\left(C^{(l)} - C^{(k)}\right)\right]^{(r)} + \left[\left(\mu^{(k)} - \mu^{(l)}\right) \cdot X^{\gamma-1}\right]^{(r)} \tag{7.2}$$

with

$$\left[2 \times \left(C^{(l)} - C^{(k)}\right)\right]^{(r)} \quad \text{and} \quad \left[\left(\mu^{(k)} - \mu^{(l)}\right) \cdot X^{\gamma-1}\right]^{(r)}$$

pre-computed for every $k$ and $l$. Then, as before we extract the $(\gamma-1)^{\text{th}}$ coefficient, to obtain $[\mathbb{d}_k - \mathbb{d}_l]$ The main drawback here is the amount of data to be pre-computed and therefore stored. However, there are two advantages to doing this. First, we apply one less external addition for every computation (although as is shown in Section 7.4.3 that is not significant time-wise). Second, and more importantly, it reduces the range that the underlying encrypted values have, therefore reducing the strain on the parameters (the B/FV operations can use a lower plaintext modulus $p$ and the TFHE operations will be more accurate).

### 7.3.2 The Tree method

At this point, we want to be able to find the index of the minimum distance. There are two ways that we can do this. The first one consists of building two parallel trees: one for the min computation and the other for a parallel argmin computation. We will call this version the "tree version".

**The** min **tree:** The min tree outputs the overall minimum distance. It compares two distances at every level and then compares the "winners" of the previous level in the current level. We start by computing an indicator $\delta$ that indicates which of two distances is the smallest one: for instance, at the first level, for the two distances $\mathbb{d}_k$ and $\mathbb{d}_l$:

$$\begin{aligned} \delta_{k,l} &= 1 \quad \text{if} \quad \mathbb{d}_k < \mathbb{d}_l \\ &= 0 \quad \text{otherwise} \end{aligned}$$

The tool used here is the bootstrap introduced in [Chi+16a]. With a sign function as output function and an addition to re-calibrate the result we obtain a scalar ciphertext of $\delta_{k,l}$[2] for any

[2]In the case where $\mathbb{d}_k = \mathbb{d}_l$ the sign bootstrap yields a random output. This is actually also the case when $\mathbb{d}_k$ is "close" to $\mathbb{d}_l$: this means the difference is in the red zone around 0 seen in Figure 5.2. See Section 7.4.1 for implications.

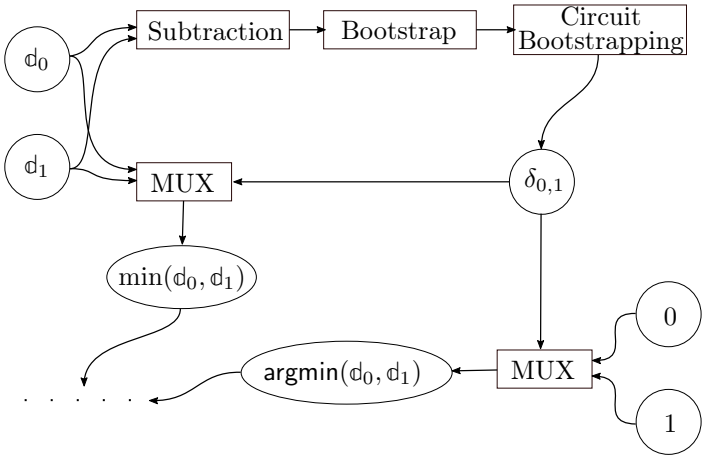


Figure 7.3: This figure shows the selection of the min and argmin for the first 2 distances. After the first round of selection, $\min(\mathbb{d}_1, \mathbb{d}_2)$ and $\min(\mathbb{d}_3, \mathbb{d}_4)$ are compared in the second round. The same goes for the argmin values. This goes on until the final argmin and min values are computed.

given $k$ and $l$. Of course 1 and 0 are not the actual torus values. In practice, for every application of the sign bootstrap, we can choose any integer $b$ so that the output is either 0 or $\frac{1}{b}$. The chosen base $b$ will be given for every use of the bootstrap operation but we may also - when convenient - only refer to the two outputs as 1 and 0. In this case for instance, we use a base $b = 4$. This $\delta$ indicator is then lifted to a TRGSW ciphertext through the circuit bootstrapping operation. We use it in the TFHE MUX gate to select the minimum distance for the next level of the tree. Per se, this means that this scheme is not fully homomorphic. Indeed every application of the MUX gate adds noise to a ciphertext that is itself reused in that same MUX gate at the next level of the tree. Therefore the parameters have to be chosen according to the depth of the tree. By repeating these operations we find ourselves, after the last comparison, with the overall minimum distance.

**The argmin tree:** The argmin tree computes the index of the minimum distance. It uses the $\delta$ values created by the min tree to select which index can go through to the next level of the tree. Every index from 0 to $d - 1$ is encoded as a polynomial representation of its own base 2 decomposition[3]: for an index $k$ it is the unique polynomial $P_k \in \mathbb{B}[X]$ such that $P_k(2) = k$. The TFHE MUX gate is applied on those indexes with the $\delta$ values as deciders. Both trees are presented in Figure 7.3.

**Potential alternate tree:** As will be shown in Section 7.4.3, this is a somewhat heavy-handed way to solve the problem. Indeed the circuit bootstrapping is the main computational burden and is only used to change the nature of the ciphertext and not to apply any "useful" operation. A way to make this tree much more efficient is to use the B/FV internal multiplication to create a B/FV MUX gate that would only consist in basically two multiplications right after the bootstrapping step. However this is not realistic as of now because of large the noise propagation in the B/FV internal multiplication. Given the output bootstrapping noise, the noise grows prohibitively after

[3]This greatly reduces the stress on the parameters induced from the MUX gate.

only one B/FV MUX gate. This could be solved with a higher precision implementation of the TFHE library, allowing for a great noise reduction from the bootstrapping operation.

### 7.3.3 The Matrix method

Another way to go about determining an argmin homomorphically is what we will informally call the "matrix" way. Instead of treating the compared distances two-by-two in a tree, we compute all of the $\delta$ values in bulk. This gives us the following matrix:

$$\begin{pmatrix} 0 & \delta_{1,0} & \cdots & \delta_{d-1,0} \\ \delta_{0,1} & 0 & \cdots & \delta_{d-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{0,d-1} & \delta_{1,d-1} & \cdots & 0 \end{pmatrix}$$

As seen in Section 7.3.2, these are obtained by subtracting the distances to compare and then applying a bootstrap operation. There are effectively $\frac{1}{2} \cdot (d^2 - d)$ bootstraps because $\delta_{i,j} = 1 - \delta_{j,i}$ for every $i, j$. At this point, we sum the lines of the matrix together and obtain:

$$\begin{pmatrix} \Delta_0 & \Delta_1 & \cdots & \Delta_{d-1} \end{pmatrix} \quad \text{where} \quad \Delta_i = \sum_{j=0}^{d-1} \delta_{i,j}$$

All of the $\Delta_i$ are between 0 and $d-1$. There is only one of them (let's call it $\Delta_{\max}$) with value $d-1$. The index for that $\Delta$ is the argmin we are looking for. Indeed, it corresponds to the only distance that is lower than every other one and therefore has $\delta$ values always equal to 1. As mentioned previously, these values of 0 and 1 are only theoretical. In practice there is a base $b$ so that the values are actually 0 and $\frac{1}{b}$. Here we set this to $2d-3$. This is important because it is the value which allows us to have $\Delta_{\max}$ be the only value above $\frac{1}{2}$ in the torus circle. As seen in Figure 7.4, this means that applying a sign bootstrap[4] on the $\Delta$ values yields a 1 only for $\Delta_{\max}$ and 0 for all other values.

**Remark:** In practice, depending on the number $d$ of distances to compare, the $\Delta$ values cannot be obtained immediately through a single sum and then a bootstrap. There are two limiting factors for the output of the operation to be correct: the noise propagation during the sum, and the precision of the sign bootstrap. Therefore, in practice, we divide the sum in "chunks" of a given number $m$ of ciphertexts and make intermediate sign bootstraps that output 1 if the intermediate sum is maximal and 0 otherwise. Therefore, the base for the first bootstrap operation has to be $\frac{1}{2m-3}$. We give the value we use for $m$ in Section 7.4.2.
This scheme is fully homomorphic because the value $m$ does not depend on the number of inputs to be compared but rather on the parameters chosen initially. Once the parameters are chosen and the value $m$ known, any number of ciphertexts to compare can be divided into chunks of $m$ ciphertexts, summed and bootstrapped, and the operation repeated because after each bootstrap operation, the noise is reinitialized. This is what makes this computation fully homomorphic. Indeed, the size of the "chunks" depends on the number of distances to compare, but not the parameters themselves.

[4] This time where $b = 4$ and where the 0 and the 1 outputs are swapped

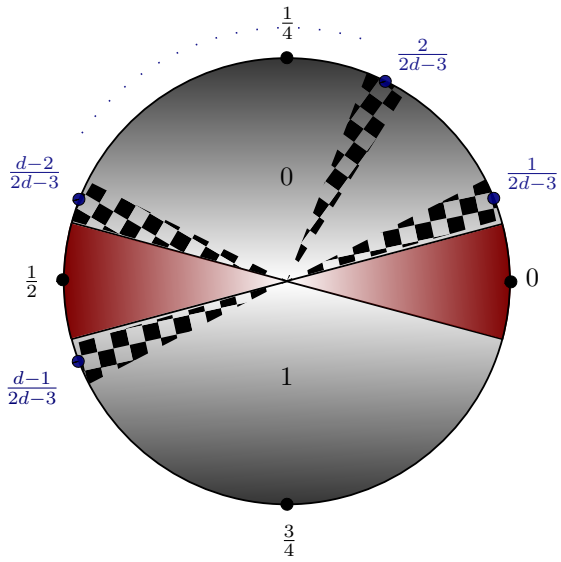


Figure 7.4: A torus representation of the $\Delta$ values. As we can see here, applying a bootstrap on all of them means only one (the highest value) will yield an output of 1. The checkered zones still represent the actual range of the message values given the error introduced in the ciphertexts. The parameters must be chosen so that they do not overlap with the red zones.

## 7.4 Practical Implementation

### 7.4.1 Precision

The two schemes presented in Sections 7.3.2 and 7.3.3 do not allow us to perform an exact homomorphic transposition of the VGGVox classifications in terms of precision. This means that the classification rate of our homomorphic system will be slightly lower than the non-encrypted one. This is due to two main factors:

- The precision of the input vectors. The output vectors of the VGGVox networks (the ones we take as inputs for our distance computations) are real numbers. Because of strains on the parameters we cannot afford to match their precision. This means we choose to round the vectors down to a certain number of digits. This strain corresponds to the fact that the values in SEAL (resp. TFHE) must not go above $\frac{p}{2}$ (resp. $\frac{1}{2}$) during the distance computation.
- The precision of the bootstrap operation. As shown in Figure 5.2, the bootstrap operation, given a set of parameters, has a "red" zone around 0 and $\frac{1}{2}$ where input values yield an uncertain output. We can arrange for the values of $\mathbb{d}_k - \mathbb{d}_l$ to never reach the zone around $\frac{1}{2}$ (by reducing the precision of the inputs as seen above). However, if two distances are too close, their difference can be too low for the bootstrap operation to yield the correct result.

Both of these sources of error are manageable. Indeed, the worst outcome is miss-classifying a speaker as the wrong one, albeit one close to the good answer. Importantly, it does not matter

if the scheme fails at computing the correct sign for distances which do not involve the final minimum distance. It will not change in any way the final result. Our only condition is that the comparisons that involve the distance which is actually the minimum be correct. We further estimate the loss in classification rate in Section 7.4.3.

There is, however, a possible source of an error that is non-avoidable, but manageable. As mentioned in Section 7.3.2, the "tree" version of the classification is levelled. This means we choose the parameters according to a given depth of our tree. If we choose our parameters poorly, the noise propagation along the tree will make the ciphertexts un-decipherable after a certain amount of computations. We show in Section 7.4.2 how to choose adequate parameters for a given depth.

### 7.4.2 Parameters

We set the parameters according to two constraints: the accuracy of the final result and the security of the overall scheme.

**Security:** We base the security of our scheme on the `lwe-estimator`. Table 7.1 shows minimum noise values for a given set of parameters. We restrict ourselves here to two security parameter values: 80 and 110. The precision for the SEAL (resp. TFHE) library is up to 7.6e−9 (resp. 1.11e−16) as of now. By this we mean the lowest value that can be taken as a standard deviation for a Gaussian sampling. Thus, any standard deviation below that could not yield an actual implementation. Therefore, we take the highest $\sigma$ of the two when a conflict occurs.

| $\lambda$ | $q$ | $N$ | $\sigma$ |
|---|---|---|---|
| 80 | 132120577 | 1024 | 2e−11 |
| 110 | 132120577 | 1024 | 9e−11 |

| $\lambda$ | $N$ | $\sigma$ |
|---|---|---|
| 80 | 1024 | 5e−13 |
| 80 | 2048 | 2e−25 |
| 110 | 1024 | 1e−9 |
| 110 | 2048 | 5e−19 |

Table 7.1: Tables presenting the security parameters for both B/FV and TFHE. In the left table, the security parameters for B/FV. In the right table, the security parameters for TFHE. $\sigma$ is the standard deviation of the Gaussian noise. Because we use both B/FV and TFHE, we need to choose the tighter security constraint between the two. The actual standard deviations chosen in the case of $N = 1024$ for both 80 and 110 security levels are therefore written in green in the tables. The ones we cannot chose are in red.

| $N$ | $p$ | $q$ | $\sigma$ |
|---|---|---|---|
| 1024 | 10752 | 132120577 | 7.6e-9 |

Table 7.2: The parameters set in the SEAL library for the initial distance computations. $N$ is the size of the initial ciphertext polynomials.

| $N_b$ | $\sigma_b$ | $B_g$ | $\ell$ |
|---|---|---|---|
| 2048 | 1.1$e$-16 | 64 | 6 |

| $N_b$ | $\sigma_b$ | $B_g$ | $\ell$ | $N_s$ | $\sigma_s$ | base | $t$ |
|---|---|---|---|---|---|---|---|
| 2048 | 1.1$e$-16 | 16 | 10 | 1024 | 5$e$-13 | 5 | 14 |

Table 7.3: Tables presenting the appropriate parameters for both the matrix and tree schemes. parameters In the upper table, the parameters set in the case of the "matrix" argmin computation. In the lower table, the parameters set in the case of the "tree" scheme. $N_b$ (resp. $N_s$) is the size of the bootstrapping key (resp. the key-switching key) polynomials and $\sigma_b$ (resp. $\sigma_s$) the standard deviation of the Gaussian noise in the bootstrapping key (resp. the key-switching key).

**Accuracy:** As mentioned in Section 7.4.1, whichever way we choose to do it, there is an inherent approximation in the homomorphic classification that we compute. In our case, we choose to use an approximation factor of 3 for the vector inputs: we truncate the values after the third digit. The parameters we use for the initial SEAL distance computation are given in Table 7.2. The value of $p$ needs to be high enough to prevent the distance values from "overflowing" and needs to verify $q = 1 \mod p$ in order to reduce noise propagation. When setting the parameters for the bootstrap and key-switch operations of the min and argmin computations, one needs to strike a good balance between efficiency and accuracy while taking security constraints into account. This leads us to the parameters presented in Table 7.3.
Furthermore, with the given parameters, in the matrix version, we can add up to 65 $\delta$ values together before applying a bootstrapping operation (as seen in Section 7.3.3).
We mentioned before that the tree scheme is levelled: this means that the given parameters here only allow us to build a tree with a certain given depth. This is not true for the matrix scheme which can, with the given parameters, be used for a classification among an arbitrary number of embedding models. For the tree scheme, the depth that these parameters allow us to go to, in this case, is 72. This means we can classify among $2^{72}$ different model embeddings. In practice, given a fixed number of models beforehand, one could reduce the size of the parameters we give here to fit the size of their tree and increase the time performance of their classification.
Overall, with these parameters, we achieve a 127-bit security in the "matrix" scheme and an 80-bit security in the "tree" scheme. In the "tree" scheme, setting parameters to obtain a higher security level would compromise the usefulness of the network in that it would greatly limit the number of comparisons we can make, or make a single comparison operation prohibitively expensive in terms of time and storage.

### 7.4.3 Performance

**Classification rate:** As we stated in Section 7.4.1, our homomorphic distance and argmin computations do not match the necessary precision to replicate exactly the classification results from the VGGVox system in [CNZ18b]. In this paragraph we evaluate how that affects the classification rate of our homomorphic equivalent given that we have only built a prototype and not fully implemented it, and therefore cannot run the classification tests directly. Given our set of parameters, we can determine that in order for the bootstrapping operation to reliably differentiate between two distances, their difference must not be greater than 2.51e−02. Below such a value, the operation will select one or the other randomly. This allowed us to simulate our homomorphic scheme in the clear by testing with several test embeddings whether, given

the shortest distance, the difference between that distance and the others was below 2.51e−02. As we saw in Section 7.4.1, it is the only condition we need for the overall argmin computation to be exact. Experimentally, we find that after truncating test vectors at the third digit (as seen in Section 7.4.2) and running the tests on several groups of embeddings, if the VGGVox system classified a recording correctly, then the smallest distance was smaller than the other distances by a wide margin (at least 0.1).

However, a more thorough evaluation of the impact of the application of our scheme on the classification performance of the system would require systematic testing that we do not provide in this work. In essence, the gap in performance when "going FHE" will depend on the precision of our argmin algorithm. For such an evaluation, look to Chapter 8, where the accuracy of our $k$-NN homomorphic scheme is evaluated. Indeed, our argmin scheme is only a specific application of our more general $k$-NN scheme.

**Timings:** With the given parameters, we are able to implement both schemes on an Intel Core i7-6600U CPU, Linux Mint 18.3 Cinnamon 64-bit, 16GiB. The overall time for a classification (the distance computations and the argmin computation) depends on the number of original reference embeddings to which we need to compare the new input.

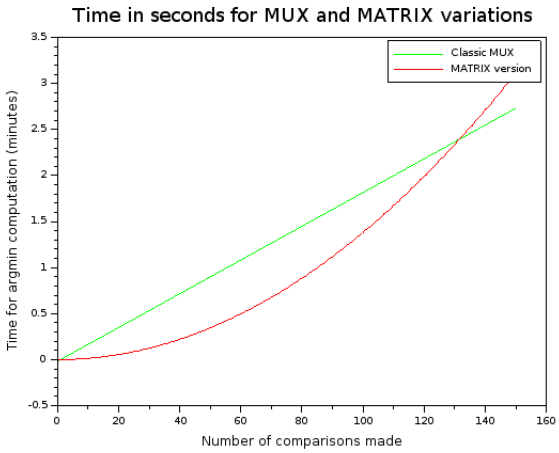


Figure 7.5: Time in minutes for tree and matrix overall classification times depending on the number of model embeddings for the VGGVox system. The time for the tree version corresponds to the green curve and the time for the matrix version to the red curve.

To be more precise, for a number $d$ of references, the tree scheme will evaluate $d-1$ comparison circuits (as seen in Figure 7.3). This means the time the tree version takes for a classification is linear in the number of model embeddings. As for the matrix scheme, it will first evaluate $(\frac{d^2-d}{2})$ bootstrap operations, and then, depending on the value of $m$ (as seen in Section 7.3.3), it will evaluate a varying number of bootstrap operations. Most importantly, the time the matrix version takes for a classification is quadratic in the number of references. Asymptotically speaking, the tree version is therefore better. This makes up for the fact that it is levelled and not fully homomorphic. Figure 7.5 shows our timing curves for both of schemes we implemented.

## 7.5 Conclusion

This work allowed us to look deeper into non-standard machine learning algorithms in search of possible applications. It also allowed us to apply the results of the Chimera framework [Bou+20] and play with several different homomorphic schemes. While our goal was met - to build an argmax homomorphic algorithm - we grew interested in the possibility to build the more general $k$-NN tool.

We can see that, on Figure 7.4, rotating the circle can allow us to have any number of values in its bottom part, and not just one like we do here. Placing every value but two in the bottom part of the circle would actually correspond to selecting the 2 closest vectors to the input vector. The same goes for any number $k$, giving us a solution to the $k$ Nearest Neighbors problem homomorphically. However, this can only work if the selection is done through one bootstrapping operation, and therefore if the number of distances is higher than $m$ (the maximum number of values we can add together before a bootstrap) then this $k$-NN scheme is not applicable. Finding a solution to this problem was the focus of the work we present in Chapter 8.

# Chapter 8

# An homomorphic k-NN classifier

**Contents**



## 8.1 Introduction

After our work on embedding-based neural networks, we found that our matrix-version algorithm could - at no additional cost - produce a $k$-Nearest Neighbor operation. However, it could do so fully homomorphically only up to a certain number of model vectors. This pushed us to work on adapting our argmin algorithm to make it into a fully homomorphic $k$-NN algorithm.

The $k$-NN method is a machine learning tool that can be used either for classification (attributing a tag/class to an unknown feature) or regression (estimating the relationship between a certain amount of variables). The principle that drives a $k$-NN classifier is that closeness of two vectors

in a Euclidean space is a good measure of class-similarity. This is of course only true for certain kinds of classification problems. For many such classification problems, $k$-NN classifiers have been found to be very efficient (compared with more recent deep learning methods for instance) and very accurate (though less so than said deep learning methods). Examples of use of $k$-NN in practical, modern applications include recommender systems (video or ad recommendation) and political analytics (voter classification).

$k$-NN classifiers have a very good accuracy given their low complexity, but suffer from poor scaling: not much in terms of accuracy is gained from increasing the model vector pool beyond a certain point, while the efficiency of the algorithm reduces. For this reason, while the strength of $k$-NN can be found in that it requires no learning phase, we decided to train our own classifiers by selecting the model vectors that most appropriately represent the rest of the model dataset - a common practice in machine learning.

In this work, we design a secure $k$-NN algorithm that can be used for both classification and regression of either a private input against a public database of neighbors or a public input against a private such database. We do so by means of homomorphic encryption techniques fine-tuned to this specific problem. As such, we thus propose an algorithm that finds the k nearest neighbors of a given vector and which is amenable to practical homomorphic evaluation timings. More precisely, the specific problem we aim to solve is known as the *$k$-NN determination problem* or the *$k$-NN problem* for short. It is the following. We are given a database of vectors that we will call *model vectors*. We are given a single vector we call the *source vector*. We aim to find the $k$ closest vectors to the source vector among the model vectors using a given norm.
We evaluate our $k$-NN determination algorithm by using it to classify in a real-world context. We then compare with existing work on the topic.

In the following, we first expose a general overview of our contribution with respect to prior work on the subject. Then comes the description of our novel $k$-NN algorithm. Finally, we detail experimental results obtained applying our algorithm on a real-world classification problem.

The work that we present here was published as [ZS21], and this chapter is based on that article.

## 8.2 Context and Motivations

### 8.2.1 Prior work

As we present prior work done on establishing confidentiality-preserving solutions for $k$-NN computations, we need to distinguish three kinds of cases depending on the nature of the database used for the $k$-NN computation. All three of them aim to protect the confidentiality of the database.

- The database is owned by a single entity and placed on a remote cloud for outsourced computation using encryption.
- The database is made up of several databases from separate entities and placed on a remote cloud for outsourced computation using encryption.
- The database is made up of several databases from separate entities that keep their own database and use a distributed computation process to obtain the final result between them and without a central server.

The third case - the distributed case - is not the topic of this work. It differs fundamentally from the first two cases in that it assumes that the database owners have computational resources at hand, and are only concerned about confidentiality. That is a legitimate case-study. However it is not our focus. We refer to [Ron+16; BD10; ZZX09; KC04; XCL06; XCL07; QA08; SKK06; ZM06; PL18] for papers which tackle this latter setup.

The first and second cases are fairly similar in their use of encryption and the outsourced nature of the computation. We will call this the *outsourced k-NN* problem. Our goal is to achieve confidentiality for the database with respect to the cloud server. Remark that achieving database confidentiality with respect to the querier involves some subtleties since, although the querier does not have access to the database, she or he can extract some information about it based on the results of its queries which she can choose freely. Additionally, [VJ10] presents an impossibility result that states that a scheme achieving private multi-client computing cannot be semantically CPA secure if it is purely a cryptographic scheme. This is a fundamental difference with the schemes that use distributed computing techniques. The interest of using encryption is in the comparatively much lower interactivity that such a scheme will incur.

Solutions in the literature vary but they all design a communication protocol between the different parties that we represent in Figure 8.1. There can be one or several *data owners*. The *data host* is the one making the $k$-NN computation. The *querier* is the entity requesting a $k$-NN computation over its source vector. In some cases, there needs to be a key distribution done by an entity we call Cryptographic Service Provider (CSP) here. Depending on the solution, there sometimes is a Trusted Computation Assistant (TCA) with access to the secret encryption key and interacting with the data host during the computation phase. We separate clearly all of these entities in Figure 8.1 in order to present a general framework for all existing schemes. However in any solution, some of those entities could be merged with others and assume several functions at once. Depending on our confidentiality requirements, we could for instance choose to have the data owner be the CSP. Several papers ([LSP15; Hu+11; Won+09; YLX13; SEJ15; Che+19]) address the outsourced $k$-NN problem and propose various solutions. We quickly review them below.

In [LSP15], the authors use an homomorphically additive encryption scheme. They use a TCA - which in this case is also the CSP - to distribute public encryption keys to the data owners and the querier while keeping the corresponding secret key. The encrypted distances for the $k$-NN computations are calculated by the data host. A kernel computation phase ($k$-NN optimization method) is made by the CSP, but it is worth noting that the distances are kept secret from it: they are masked by the data host before being sent to the CSP. The comparison and classification phases occur with interactions between the CSP and the DH. They use a garbled circuit (introduced in [Yao86]) for this.

In [Hu+11], the authors build a system in which there is no data host, and the data is given to the querier in encrypted form. Then the queriers interact with a TCA (which has a decryption algorithm) to extract the result. One of the drawbacks of this scheme is the fact that the querier (called client in [Hu+11]) learns some incomplete plaintext information on the data from the TCA. In [YLX13], the authors use this information to design an attack in the CPA model, in which the client can eventually obtain the actual data values in plaintext in the case of an honest TCA with no collusion. Because of this attack, we will not include this scheme into our comparison table (Table 8.1).

In [SEJ15], the authors propose a secure multiparty $k$-NN computation. The data owner outsources its encrypted database to the DH and its secret key to a TCA. Then the querier interacts

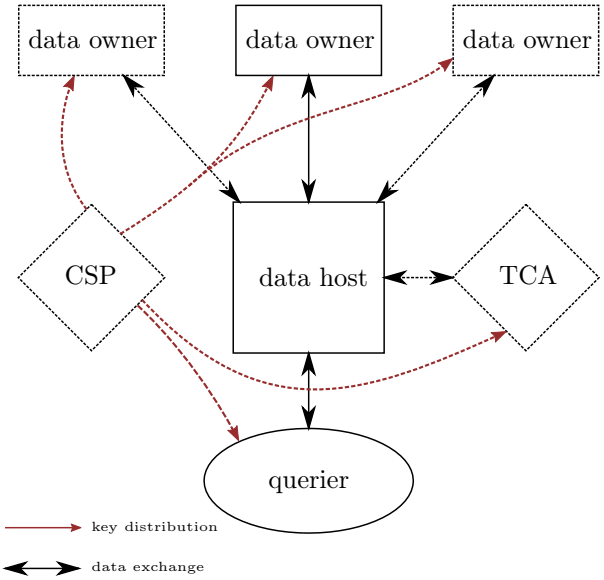


Figure 8.1: A general design for protocols in the literature that tackle the *outsourced* problem. The data owner, data host and querier are always present and are drawn in full lines while other, "optional", entities are drawn with dashed lines. TCA: Trusted Computation Assistant. CSP: Cryptographic Service Provider.

with both of them to complete the $k$-NN determination in linear time.

In [Won+09], the authors use what they call an "asymmetric scalar-product-preserving encryption". This self-explanatory type of encryption allows them to do away with a TCA. The queries and the database are encrypted using two different encryption algorithms but with the same key. Then the data host can compute the comparisons between every distance. The k encrypted data points that are closest to the query can then be sent back to the querier for decryption. This means the data host has access to the scalar products $p \cdot q$ for every data point $p$ and every query $q$ sent by the querier. Therefore, in a CPA attack model, with the data host as an attacker, it can choose the queries $q$ and [YLX13] shows that it can recover the data points through a set of linear equations (as many as there are queries). However, the data host can only mount a CPA attack with the help of the querier (it needs an encryption key which is secret). This means that this scheme is not resistant to collusion between the data host and the querier. This is actually the case of all other encryption-based schemes we encountered in the literature. It is also the case for our scheme.

One exception is that of the scheme in [YLX13]. They only circumvent this problem by providing the data host with encryptions of portions of the database. The final result will not be an actual nearest neighbor but rather the nearest partition. This scheme is structurally different from all other schemes we mention here because it solves another problem in order to circumvent this structural security flaw. This is why we will not be comparing our scheme to it.

In [Che+19], the authors design a secure $k$-NN scheme based on both additive homomorphic encryption and garbled circuits (therefore requiring interactions between the data host and the querier). They provide a secure implementation of two $k$-NN algorithms: a linear scan algorithm

which has a linear complexity; and a clustering-based algorithm which has a sub-linear complexity but does not aim to output the exact $k$ nearest neighbours.

Table 8.1 compares the schemes we presented from the literature with our own scheme with respect to the two most important attributes for a secure $k$-NN determination scheme: its non-interactivity and its complexity. The schemes mentioned here diverge in a lot of ways. Most notably, in the way that they implement (or not) specific optimizations to the $k$-NN algorithm (kernels for instance) or whether they implement a secure majority-class voting at the end. We chose to focus on the basic $k$-NN algorithm. As a general rule we can see that our scheme, compared with pre-existing ones, trades non-interactivity with a higher complexity. This makes our scheme asymptotically slower (though still practical for real-world problems as is shown in Section 8.4.4) but requires no interaction for the $k$-NN computation. All other schemes require some sort of interaction.

One exception is [Won+09] which is both linear in complexity and works offline. However that scheme provides a lower level of security: the use of their scalar-product-preserving encryption means the data host will have access to the distance values in the clear. We propose a scheme which guarantees (to the level of the security of the underlying encryption scheme) that no information will leak during the $k$-NN computation.

| | this work | [LSP15] | [SEJ15] | [Won+09] | [Che+19] |
|---|---|---|---|---|---|
| Non-Interactive | ✓ | ✗ | ✗ | ✓ | ✗ |
| Complexity | $x^2$ | $x$ | $x$ | $x$ | $x$ |

Table 8.1: This table shows a comparison of the two most important attributes for a secure $k$-NN scheme: its non-interactivity and its complexity. The complexity of the corresponding scheme is linear when $x$ is used, and quadratic when $x^2$ is used. As mentioned, [Che+19] introduces two secure $k$-NN algorithms and one of them is sub-linear. However this clustering-based algorithm does not output the exact result.

### 8.2.2 Our Contribution

In this work, we propose a fully homomorphic $k$-NN algorithm. This algorithm works in two settings: either the database is encrypted and the query is clear; or the query is encrypted and the database is clear. Achieving encryption for both the query and the database is within reach but not yet attainable due to a high noise propagation issue. These two "settings" allow us to design two protocols for secure $k$-NN determination where the computation itself is fully non-interactive. Figure 8.2 represents the two protocols for a secure $k$-NN computation that our scheme allows for. In both protocols, the confidentiality of both the query and the database is achieved. What changes is who needs to encrypt their data and who is responsible for handling the computation.

The *encrypted query* protocol has the query $q$ be sent encrypted by the querier, to the data owner (which is in this case also the data host) using a bracket notation for encrypted data. The data owner then applies our secure $k$-NN algorithm over an encrypted query and a clear database and sends the encrypted result $r$ back to the querier.

The *encrypted database* protocol requires the encrypted result $[r]$ computed by the data host to be sent back to the querier where the querier randomizes it (with an addition of a random vector $R$) using the data owner's public key: $[r + R]$. Then the masked encrypted result is sent to the

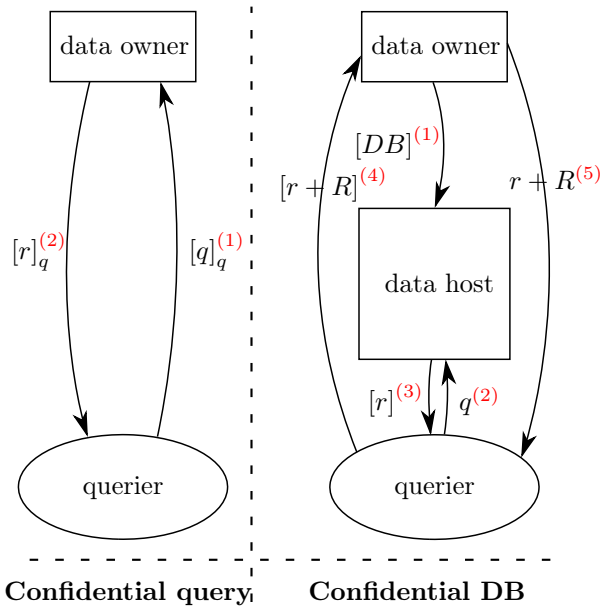


Figure 8.2: The two protocols for a secure $k$-NN computation that our scheme allows for. We use simple brackets ($[\cdot]$) to indicate that data is encrypted by the data owner and brackets with an index $q$ ($[\cdot]_q$) to indicate that data is encrypted by the querier. The order in which the data is exchanged is indicated in red: $q^{(2)}$ means $q$ is the second data exchange to happen in the protocol. The interactions between the querier and the data host occur after the computation is completed.

data owner for decryption, which yields a clear masked result $(r + R)$ that is sent back to the querier so that it can recover the actual result $r$.

**Remark.** The encrypted database protocol can be changed so that the data owner does not have to decrypt the final result, thereby eliminating interactivity even after the computation is done. This change requires the data owner to encrypt its database with the querier's public key and then send its encrypted database to the data host. The data host can then compute the result with a clear query from the querier and send it back to the querier. However this protocol leaves the data owner highly susceptible to collusion between the data host and the querier.

We show that we can implement both of these protocols with quadratic complexity with respect to the size of the database but constant with respect to $k$. We show that they achieve close to optimal classification rate over a real-world classification problem. Over this classification problem our scheme is very efficient, and only reaches prohibitive timing figures over much bigger databases.

### 8.2.3 Scenario and threat model

This section aims at pointing out the kind of classification use cases that can be meaningfully addressed by the ability to run a $k$-NN classifier in the homomorphic domain. We do so in an

abstract fashion, independently of the underlying FHE techniques (which are duly detailed in the sequel). We start by considering the concrete e-health scenario in which a doctor (*operator*) wishes to benefit from a diagnosis service provided by a laboratory (*server*) for one of its patient (*user*) on a given class of pathologies. The laboratory is able to provide this service thanks to a highly sensitive database of previous cases which it cannot share with third parties (due to either or both legal constraints or commercial value) and which it uses at the core of a $k$-NN classifier. In this context, a confidentiality-preserving diagnosis service can work as follows. The doctor is the owner of an FHE private/public keys pair. The patient then uses the doctor's public key to encrypt its data which it then sends to the laboratory. The lab evaluates the $k$-NN classification of the (encrypted) patient data in the homomorphic domain against its database (which is in clear form) and produces an (encrypted) classification. The latter (encrypted) classification is then sent to the doctor for decryption and interpretation. In terms of confidentiality, this kind of protocol has several desirable properties. The raw patient data remains private from the lab (as it sees and manipulates them only in encrypted form) and the doctor (as it never gets them and is only granted access to the resulting classification). So, with respect to its raw data confidentiality, the patient only has to trust that the doctor and the laboratory do not collude. Additionally, the laboratory's precious database of past cases remains on its own server and its confidentiality is also preserved from the doctor (and the patient, which is most likely to also learn its own classification or a byproduct of it). It should however be emphasized that, although this architecture offers provable security on the patient's raw data with respect to confidentiality threats coming for the laboratory, the guarantees it offers on the confidentiality of both the raw patient data and the laboratory's database from threats coming from the doctor are less formal as the very fact that the doctor has access to a cleartext classification leads to some (useful) leakage on the raw patient data and may also induce some leakage on the database if the doctor and patient collude. This latter leakage is inherent to the service provided and, if not acceptable, can be mitigated by other countermeasures such as noising the output of the classifier (differential privacy gives us the theoretical toolbox to do so meaningfully). This countermeasure is hard to consider in serious use-cases such as medical ones, where patient health - and therefore classification accuracy - cannot be traded for database confidentiality. Another possible countermeasure would be throttling the request rate, which can be performed by a given entity. In some cases, it is also possible for the doctor and the patient to be the "same" entity e.g., in the case of a doctor entering the data provided by his or her patient during a consultation. Additionally, the above illustrative example potentially applies in many other user/server/operator situations.

Beyond scenarios in which a private (encrypted) request is run over a cleartext database, the case where a cleartext request is run over a private (encrypted) database is also of practical relevance. To stay in the medical field, we can consider the case of a pharmaceutical company (*operator*) which needs to run an epidemiological study over a database of cases stored by one or more hospitals (*server*s). For normative and legal reasons, as is usual in the medical field, the data owned by a hospital cannot leave its information system. Yet, the pharmaceutical company does not wish to reveal the nature of the study it will perform by means of a $k$-NN classifier i.e. it does not wish to reveal its database of reference vectors to the hospital. In this context, the firm can then send its database of reference vectors encrypted under its own FHE public key and the hospital can then homomorphically evaluate the $k$-NN over its database of patient data (which it has access to in clear form), sending the (encrypted) classifications back to the firm for further analysis once processing complete (note that, in this second scenario of batch processing of a possibly large database under mild latency constraints, the non-interactivity of our approach is highly desirable to avoid unnecessary resource-wasting synchronization during the processing).

In this context, the raw hospital data are not disclosed to the pharmaceutical company and stay on premise. Additionally, the sensitive data from the firm is not disclosed to the hospital. The above illustrative example also applies in many operator/server situations (e.g. cyber-attack threat monitoring with private criteria in cleartext traffic, recognition of private faces in cleartext video streams [ISZ19], ...).

Lastly, note that our solution does not allow to address the case where both the request and the database are encrypted. We believe that adapting our solutions to allow for the encryptions of both of them will be possible soon. It would require an increased precision in the underlying FHE library we used for implementation and would come at, as of yet unknown, performance costs.

## 8.3 Our k-NN algorithm

In this section, we will present our general algorithm solving the $k$-NN problem over encrypted inputs. However, for simplicity sake, we will first present it with all inputs and outputs as clear values. We show in section 8.3.5 how and why this algorithm can be used efficiently in a fully homomorphic setting. First of all, we assume that the integer $k$ (as in $k$-NN) is set.

### 8.3.1 Problem definition

The $k$-NN problem aims to find the k Nearest Neighbors of a given vector (the source vector) among a number of vectors (the model vectors).
The problem that we aim to solve is to have the data host perform a $k$-NN determination over an encrypted set of reference vectors, non-interactively. It is given encryptions of $d$ real model vectors $c^{(i)} \in \mathbb{R}^\gamma, i \in \{1, d\}$ of dimension $\gamma$ by the data owner. The data owner also gives it two bootstrapping keys which are used in TFHE computations and cannot be used to retrieve encrypted data without a decryption key. The querier sends it its query: one clear source vector $m \in \mathbb{R}^\gamma$. Again, it could very well be the source vector that is hidden under a layer of encryption and the model vectors that are clear. See Figure 8.2 for a presentation of the two protocols. It is made evident in section 8.3.5 that the distance computation can be done in the same way in either case and the rest of the computation is identical.
The data host's goal is to obtain $d$ encrypted binary values - one for every model vector - with a 1 for every model vector that is among the $k$ closest vectors to the source and a 0 for every model vector that is not. The results of the $k$-NN computation need to have a correct decryption with overwhelming probability.

### 8.3.2 Distance computation

What interest us are not actually the distances between the model vectors and the source vectors themselves but rather a comparison of distances. In fact we are going to compute the difference of the squares of distances. This means that if $\mathbb{d}_i$ is the distance between vector $i$ and the source for a given $i \in \{1, d\}$, then we want to compute $\mathbb{d}_i^2 - \mathbb{d}_j^2$ for every $i, j \in \{1, d\}$.

### 8.3.3 Delta Values

For all $i, j$ in $\{1, d\}$, we compute the sign of the difference of the two squared distances $\mathbb{d}_i^2, \mathbb{d}_j^2$. As in Chapter 7, we obtain our $\delta$ matrix by computing:

$$\begin{aligned} \delta_{i,j} &= 1 \quad \text{if} \quad \mathbb{d}_i < \mathbb{d}_j \\ &= 0 \quad \text{otherwise} \end{aligned}$$

The matrix:

$$\begin{pmatrix} 0 & \delta_{1,2} & \cdots & \delta_{1,d} \\ \delta_{2,1} & 0 & \cdots & \delta_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{d,1} & \delta_{d,2} & \cdots & 0 \end{pmatrix}$$

### 8.3.4 The scoring algorithm

From this point on, we have to diverge from the algorithm in Chapter 7. Indeed, we recall that given a set of parameters, we were limited to a fixed number of sums $m$ before we need to apply a bootstrapping operation.

We set an index $i$ for one of the model vectors, and aim to produce a 0 value if that vector is not among the k nearest neighbors to the source and a value of 1 if it is. We consider that $k < m$. We will see in section 8.4.3 that this will always be the case in practice.

We introduce a *scoring operation* to solve the $k$-NN problem over an arbitrary amount of model vectors. Figure 8.3 presents the algorithm applied in the first phase of our $k$-NN computation: over the first $m$ values in column $i$. The algorithm takes $m$ inputs and has one output. Its building blocks are the summing operation and `Sign` operations which apply the following transformation:

$$\begin{aligned} \texttt{Sign}_l : \mathbb{N} &\to \mathbb{B} \\ x &\mapsto 0 \quad \text{if } x \leq m - l \\ &\quad\ 1 \quad \text{if } x > m - l \end{aligned}$$

We can formalize our scoring operation as the following function $\mathcal{S}_{k,m} : \mathbb{N}^n \to \{0, \ldots, k\}$:

$$\begin{aligned} \mathcal{S}_{k,m} : (x_1, \ldots, x_n) &\mapsto \max \left( 0 \ , \ k - m + \sum_{l=1}^{n} x_l \right) \\ &\quad \varnothing \quad \text{if } \sum_{l=1}^{n} x_l > m \end{aligned}$$

For the output to be in $\{0, \ldots, k\}$, we need to have $\sum_{l=1}^{n} x_l \leq m$. This corresponds to the condition that lead us to design this algorithm.

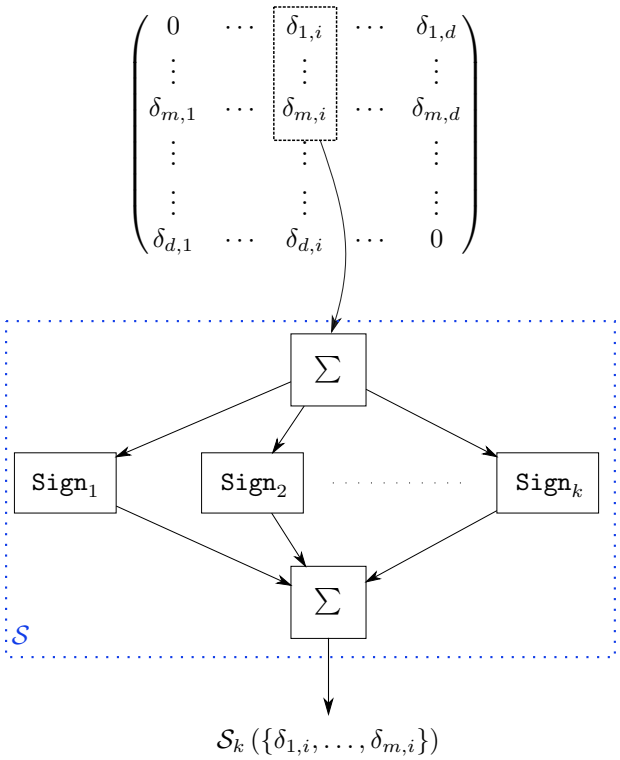


Figure 8.3: A figure presenting our scoring algorithm. Here it is presented parsing through the first $m$ values in one column - the first phase of the $k$-NN computation. The values $\delta_{1,i}, \ldots, \delta_{m,i}$ are summed and then their sum is parsed by the `Sign` operations. The $k$ outputs are summed again at the end. The operation - the dotted blue box called $\mathcal{S}$ - outputs a score between 0 and $k$ we write $\mathcal{S}_{k,m}(\{\delta_{1,i}, \ldots, \delta_{m,i}\})$.

When applied to binary inputs ($\mathbb{B}^m$), this operation counts the number of 0s and returns either $(k - \#0)$ or 0. An example, for $m = 7$ and $k = 3$:

$$\mathcal{S}_{3,7}(0,1,1,1,0,0,1) = \max(0, k-3) = 0$$
$$\mathcal{S}_{3,7}(1,0,1,1,1,1,0) = \max(0, k-2) = 1$$
$$\mathcal{S}_{3,7}(1,1,0,1,1,1,1) = \max(0, k-1) = 2$$

In our $k$-NN computation, we apply it on the $\delta$ values from the matrix, which are binary values. And if $\delta_{j,i} = 0$, that means that vector $j$ is closer than vector $i$ to the source. Therefore the number of 0s in our $i^{\text{th}}$ column counts the number of vectors that are closest to the source than vector $i$. This is why we call it a scoring operation. The higher the score given by our operation, the closer vector $i$ is to the source. The operation, in an FHE setting, is still limited to inputs of sum lower than $m$. The output is in the range $\{0, \ldots, k\}$. Therefore, since $k < m$, after applying the scoring operation once, there is "room" to add $m-k$ more values to the output and applying the operation again. Even if the output were 0, in an FHE setting, that information is

not known, therefore we have to plan everything as if the output value were its maximum value: $k$. Let's formalize this with a proposition.

**Proposition 1.** *Let* $x_1, \ldots, x_{2m-k} \in \mathbb{B}$.
*Let A denote:*

$$\mathcal{S}_{k,m}\left(x_1, \ldots, x_{m-k},\, \mathcal{S}_{k,m}\left(x_{m-k+1}, \ldots, x_{2m-k}\right)\right)$$

*Let B denote:*

$$\mathcal{S}_{k,2m-k}\left(x_1, \ldots, x_{2m-k}\right)$$

*in an FHE setting, we cannot compute B directly because it would require us to sum more than m binary values. However we can compute A and:*

$$A = B$$

*proof of Proposition 1.* There are two cases to look at.

**Case 1:** $\displaystyle\sum_{l=m-k+1}^{2m-k} x_l \leq m - k$

then we have

$$\mathcal{S}_{k,m}\left(x_{m-k+1}, \ldots, x_{2m-k}\right) = 0$$

and since $x_1, \ldots, x_{2m-k} \in \mathbb{B}$, then $\sum_{l=1}^{m-k} x_l \leq m - k$.
Therefore $A = 0$. Also,

$$\begin{aligned}\sum_{l=1}^{2m-k} x_l &= \sum_{l=1}^{m-k} x_l + \sum_{l=m-k+1}^{2m-k} x_l \\ &\leq m - k + m - k \\ &\leq 2m - 2k\end{aligned}$$

And we have:

$$B = \max\left(0\ ,\ 2k - 2m + \sum_{l=1}^{2m-k} x_l\right)$$

Therefore $B = 0 = A$.

**Case 2:** $\displaystyle\sum_{l=m-k+1}^{2m-k} x_l > m - k$

Then,

$$\mathcal{S}_{k,m}\left(x_{m-k+1}, \ldots, x_{2m-k}\right) = k - m + \sum_{l=m-k+1}^{2m-k} x_l$$

And

$$\begin{aligned} A &= \mathcal{S}_{k,m}\left(x_1, \ldots, x_{m-k},\ k-m+\sum_{l=m-k+1}^{2m-k} x_l\right) \\ &= \max\left(0,\ k-m+\sum_{l=1}^{m-k} x_l + k - m + \sum_{l=m-k+1}^{2m-k} x_l\right) \\ &= \max\left(0,\ k-(2m-k)+\sum_{l=1}^{2m-k} x_l\right) \\ &= \mathcal{S}_{k,2m-k}\left(x_1, \ldots, x_{2m-k}\right) \\ &= B \end{aligned}$$

□

Therefore, by iterating enough times the scoring operation, we can obtain the score of a set of any size. This means we can compute

$$\mathcal{S}_{k,d}\left(\delta_{1,i}, \ldots, \delta_{i-1,i}, \delta_{i+1,i}, \ldots, \delta_d\right) \tag{8.1}$$

Which is equal to 0 if vector $i$ is not among the $k$ nearest neighbors and a non-zero value if it is one of the $k$ closest vectors to the source. Note that we removed $\delta_{i,i}$ from the computation as its value is trivial and known to be 0. In fact, (8.1) is not exactly the value that we are looking for. We want a constant output value of 1 for the $k$ closest vectors. This means the last scoring operation is replaced with a sum and the $\texttt{Sign}_k$ operation. The full algorithm is represented in Figure 8.4.

Furthermore, if - for the last sum - there are less than $m-k$ values left in matrix column, then for the last $\texttt{Sign}_k$ operation to output the correct value, we need to pad the sum with an appropriate value.

### An example execution

An example of our $k$-NN algorithm applied in the case of $d = 7$ model vectors is given in Figure 8.5. We are looking for the $k = 2$ closest vectors to the source vector. We can only add $m = 4$ $\delta$ values at a time. The model vectors are represented in a drawing on top in their Euclidean space as small stars, and the source vector is the larger star. As we can see from the figure, the two closest vectors are vectors 2 and 7. We do not go into the details of the distance computation and the distance comparisons that go into creating the associated $\delta$ matrix. We show how to apply the algorithm on the $5^{\text{th}}$ vector. From the drawing, we can see that it is the $3^{\text{rd}}$ closest vector and therefore we should obtain a result of 0. The matrix diagonal is "removed" in the sense that we do not take it into account. We take the first $m = 4$ $\delta$ values for the $5^{\text{th}}$ column and make them go through the scoring algorithm $\mathcal{S}$. Since there is one 0 among those first values, the output of the scoring algorithm at this point is $k - 1 = 1$. The output of the scoring algorithm is at most of size $k = 2$. Since we do not know its value (in an FHE setting), we can only add $m - k = 2$ more $\delta$ values to it (we cannot have a sum go above the value $m$).
Therefore, the rest of the algorithm consists of adding 2 fresh values to the output of the last

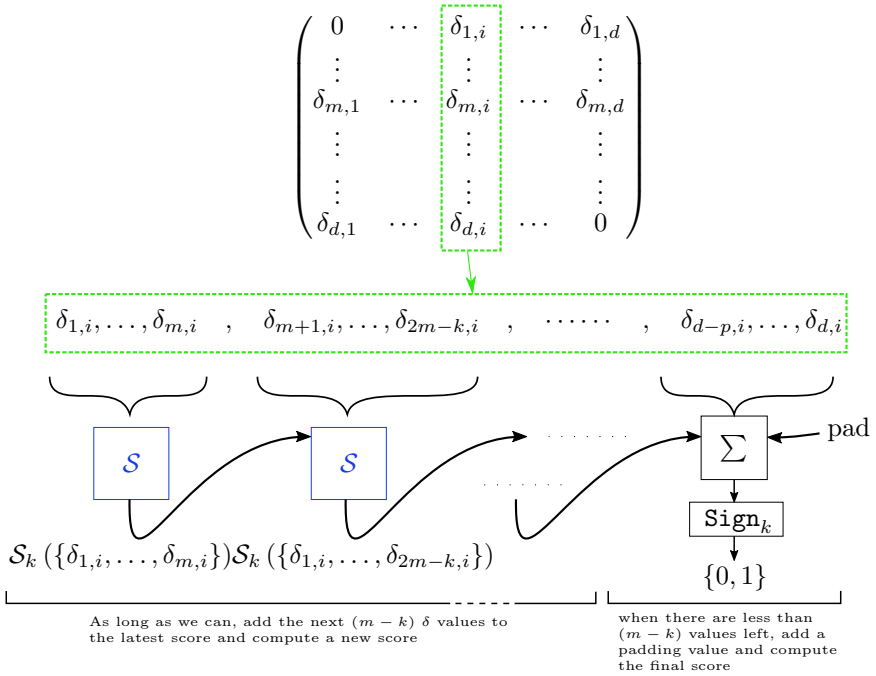


Figure 8.4: A figure representing the final score computation for a given vector $i$. The blue $\mathcal{S}$ boxes correspond to the scoring algorithm as represented in the dotted blue box in Figure 8.3. It is important to note that for simplicity sake, we did not make it clear in the figure that we remove $\delta_{i,i}$, the diagonal value, from the computation. At the last step, a padding (pad) is added to the sum to obtain the correct final result. That padding is pad $= \max(0, m-k-p-1)$.

scoring operation and running a new scoring operation. Here we are left with only two values, therefore we add them with the output of the first $\mathcal{S}$ box and compute the score. We find an overall score of 0, which means that the $5^{\text{th}}$ vector is not among the 2 closest. We do not know anything else about it.
If the matrix was of size 6, then we would add a $7^{\text{th}}$ line of 1s to the matrix as padding. Since the algorithm scores according to the number of 0s, this would not affect the overall score but it would allow us to have the right number of inputs for the scoring algorithm. In the context of an FHE computation, this padding would be added as plaintext values and therefore induce no noise.

### 8.3.5 Going FHE

From the start of Section 8.3, we have so far presented a $k$-NN algorithm that works over clear values. To apply our algorithm in an FHE setting we need to use the equivalent homomorphic operations on encrypted inputs.

#### Encoding and Encryption

We use the TFHE encryption scheme and therefore will be all values to torus values in $\mathbb{T}$. In this scheme, we only need to define three rescaling bases: one as a common encoding base for all the

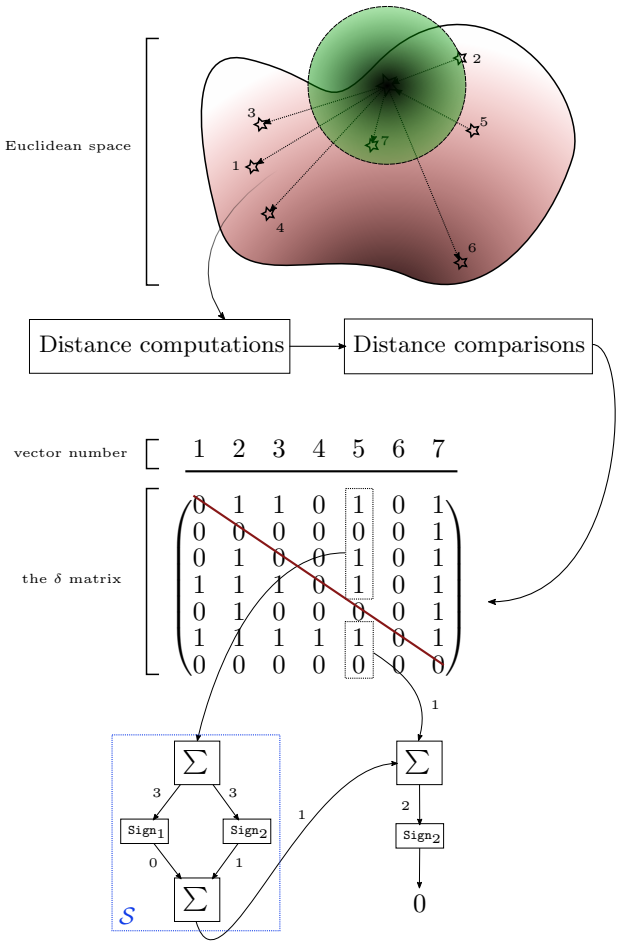


Figure 8.5: An example of our $k$-NN algorithm applied in the case of $d = 7$ model vectors. We are looking for the $k = 2$ closest vectors to the source vector. We can only add $m = 4$ $\delta$ values at a time.

model vectors and the source vector: $\nu$; one as a common encoding base for all the $\delta$ values: $b_\delta$; one as the encoding base for the final output values: $b_\mathrm{f}$.

Therefore, to take an example, a $\delta$ value that we presented in section 8.3.3 as being equal to either 0 or 1 will actually be equal to either 0 or $\frac{1}{b_\delta}$. And a sum of $m$ such values will be in $\{0, \frac{1}{b_\delta}, \ldots, \frac{m}{b_\delta}\}$.

### Squared distance difference computation

Every squared distance is computed between an encrypted and a non-encrypted vector as explained in section 8.3.1. Again we assume that the source vector is plain and the model vectors are encrypted. The opposite yields a strictly equivalent way to compute the distances and from then on, everything is encrypted and therefore nothing changes. The encrypted distance computation uses exclusively TFHE ciphertexts but is otherwise very similar to the one presented

in Chapter 7. We do describe it in detail however, to better present the impact of the different rescalings and approximations we make to the initial data values.

We are given $d$ real model vectors: $c^{(i)} \in \mathbb{R}^\gamma, i \in \{1, \dots, d\}$ of dimension $\gamma$. However, we encode these vectors as torus vectors in order to have them be encrypted in the TFHE encryption scheme. Therefore we define a base $\nu > \max_i \left(\left\|c^{(i)}\right\|_\infty\right)$ and rescale every vector by $\nu$ to obtain torus vectors.

We call $\mu \in \mathbb{R}^\gamma$ the source vector. First of all, it needs to be rescaled using the same value used for the model vectors: $\nu$. Secondly, when going FHE, we have to run the $k$-NN computation on an integer source vector and not a real one. However $\mu$ is a real data vector and therefore needs to be rounded to an integer vector after a scaling is performed to preserve the precision. Therefore, with an additional rescaling factor $\tau \in \mathbb{N}$, we actually have to transform every $\mu_i$ into $\lfloor \frac{\tau \times \mu_i}{\nu} \rceil \in \mathbb{N}$. We then encode this rounded and rescaled source vector as a polynomial:

$$M = \sum_{l=0}^{\gamma-1} \left\lfloor \frac{\tau \times \mu_{\gamma-l}}{\nu} \right\rceil \cdot X^l$$

However, introducing this $\tau$ factor by itself would change the distance computation result. Therefore, we need to rescale the model vectors by $\tau$ as well. Encoding the twice-rescaled model vectors a polynomials gives us: $\forall i \in \{1, d\}$,

$$C^{(i)} = \sum_{l=0}^{\gamma-1} \frac{c_{l+1}^{(i)}}{\tau\nu} \cdot X^l$$

We are actually given an encryption $\left[C^{(i)}\right]^{(r)}$ for every $i$. And we want to obtain encryptions of every differences of every distances between vectors $c^{(i)}$ and $\mu$. In other words, if we write $\mathbb{d}_i$ the distance from $\mu$ to $c^{(i)}$ for every $i$, then we want to obtain $\left[\frac{1}{\nu^2}(\mathbb{d}_i^2 - \mathbb{d}_j^2)\right]$ for every $i \neq j$.

In order to simplify the computation, we assume that the party that encrypted the $\left[C^{(i)}\right]^{(r)}$ ciphertexts also pre-computed and encrypted:

$$A_i = \left(\sum_{l=1}^{\gamma} \left(\frac{c_l^{(i)}}{\nu}\right)^2\right) \cdot X^{\gamma-1}$$

for every $i$.

At this point, given M, $\left[C^{(i)}\right]^{(r)}$ and $[A_i]^{(r)}$ we can compute:

$$2M \cdot \left(\left[C^{(j)}\right]^{(r)} - \left[C^{(i)}\right]^{(r)}\right) + [A_i]^{(r)} - [A_j]^{(r)} \tag{8.2}$$

**Proposition 2.** *For $\tau$ a high enough power of 10, if we extract the $(\gamma-1)^{th}$ coefficient from (8.2), we obtain $\left[\frac{1}{\nu^2}(\mathbb{d}_i^2 - \mathbb{d}_j^2)\right]$.*

*proof of Proposition 2.* If we assume that the homomorphic operations work as expected (this depends only on an appropriate choice for the parameters) then we can simply show that the same computation, but over clear data, yields the right result.
We have:

$$2M \cdot C^{(i)} = 2\left(\sum_{l=0}^{\gamma-1} \lfloor \frac{\tau \times \mu_{\gamma-l}}{\nu} \rceil \cdot X^l\right) \cdot \left(\sum_{l=0}^{\gamma-1} \frac{c_{l+1}^{(i)}}{\tau\nu} \cdot X^l\right)$$

Therefore its $(\gamma-1)^{\text{th}}$ coefficient is $2\times\sum_{l=1}^{\gamma} \lfloor \frac{\tau\times\mu_l}{\nu} \rceil \times \frac{c_l^{(i)}}{\tau\nu}$. If $\tau$ is high enough, then $\lfloor \frac{\tau\times\mu_l}{\nu} \rceil = \frac{\tau\times\mu_l}{\nu}$ and this means the $(\gamma-1)^{\text{th}}$ coefficient of

$$A_i + \left(\sum_{l=1}^{\gamma} \frac{\mu_l^2}{\nu^2}\right) X^{\gamma-1} - 2M \cdot C^{(i)}$$

is

$$\begin{aligned}
&\sum_{l=1}^{\gamma} \left(\frac{c_l^{(i)}}{\nu}\right)^2 + \sum_{l=1}^{\gamma} \frac{\mu_l^2}{\nu^2} - 2 \times \sum_{l=1}^{\gamma} \frac{\mu_l c_l^{(i)}}{\nu^2} \\
=&\frac{1}{\nu^2} \sum_{l=1}^{\gamma} \mu_l^2 - 2\mu_l c_l^{(i)} + \left(c_l^{(i)}\right)^2 \\
=&\frac{1}{\nu^2} \sum_{l=1}^{\gamma} \left(\mu_l - c_l^{(i)}\right)^2 \\
=&\frac{1}{\nu^2} \left\|\mu - c^{(i)}\right\|_2^2 = \frac{1}{\nu^2} \mathbb{d}_i^2
\end{aligned}$$

Therefore the $(\gamma-1)^{\text{th}}$ coefficient of

$$\begin{aligned}
&A_i + \cancel{\left(\sum_{l=1}^{\gamma} \frac{\mu_l^2}{\nu^2}\right) X^{\gamma-1}} - 2M \cdot C^{(i)} \\
&- \left[A_j + \cancel{\left(\sum_{l=1}^{\gamma} \frac{\mu_l^2}{\nu^2}\right) X^{\gamma-1}} - 2M \cdot C^{(j)}\right] \\
=&A_i - A_j + 2M \cdot \left(C^{(j)} - C^{(i)}\right)
\end{aligned}$$

is $\frac{1}{\nu^2}(\mathbb{d}_i^2 - \mathbb{d}_j^2)$. □

The value of $\tau$ will determine the extent to which the result is approximated. In section 8.4.3, we can see that experimentally, allowing some approximation does not impact the overall $k$-NN result significantly.

This homomorphic computation can only work if the difference does not go above $\frac{1}{2}$ or below $-\frac{1}{2}$. Indeed, we will consider a value between 0 and $\frac{1}{2}$ to be positive and one between $-\frac{1}{2}$ and 0 to be negative. Therefore overflowing in the difference computation will cause us to return a wrong sign value. This requires choosing a base value $\nu$ appropriate for the given data range. We explain our choices for the different base values in section 8.4.1.

### Delta computation

At this point, we want to obtain the sign of the differences $\left[\mathbb{d}_i^2 - \mathbb{d}_j^2\right]$ for every $i \neq j$ (it is the same as that of $\left[\frac{1}{\nu^2}(\mathbb{d}_i^2 - \mathbb{d}_j^2)\right]$). We do this using the bootstrapping operation. The output of the sign bootstrapping operation applied to $\left[\frac{1}{\nu^2}(\mathbb{d}_i^2 - \mathbb{d}_j^2)\right]$ is $[\delta_{i,j}]$. As mentioned in section 8.3.5, the output values will either be 0 or $\frac{1}{b_\delta}$ with $b_\delta$ a given base. We do not actually compute all of the $\delta$ values as $\delta_{i,j} = 1 - \delta_{j,i}$ for every $i, j$. Therefore, there are only $\frac{1}{2} \cdot (d^2 - d)$ bootstrapping operations.

### The scoring operation

Figure 8.6 is a representation of the scoring operation applied to torus variables. To represent any variable, we present it in the figure as every value it can take in the torus: 0 and $\frac{1}{b_\delta}$ for $\delta$ variables for instances. The scoring operation starts with a sum of $\delta$ variables. The sum of ciphertexts is a standard operation. As shown in Figure 8.6, the resulting ciphertext can hold one of $m+1$ different values from 0 to $\frac{m}{b_\delta}$. We want the greatest possible value to be $\frac{1}{2}$ and therefore we need to set $b_\delta = 2m$. From there intuitively, we apply $k$ rotations to obtain $k$ ciphertexts on which we run the sign bootstrapping operation.The sign bootstrapping is usually represented as giving an output of 0 for negative values. For this bootstrapping operation, we still apply a sign operation but invert the outputs so that our bootstrapping outputs $\frac{1}{b_\delta}$ for "negative" values. The values are not actually seen as negative here but rather as *overflowing* from the rotation we apply. We can see in Figure 8.6 that only the greatest possible value will yield an output of $\frac{1}{b_\delta}$ over a sign bootstrapping of the first rotated ciphertext. Only the $l$ greatest possible values will yield an output of $\frac{1}{b_\delta}$ over a sign bootstrapping of the $l^{\text{th}}$ rotated ciphertext.

This figure therefore represents the homomorphic equivalent of the `Sign` operations. However, though this is exactly how these `Sign` operations work intuitively, their implementations are slightly more subtle. The homomorphic operations $\texttt{Sign}_1, \ldots, \texttt{Sign}_k$ are all computed together at the same time on the same input, and can be seen as one single operation with one input and $k$ outputs. We can do this by applying a sign bootstrapping over the first rotated ciphertext and then extracting $k$ different outputs where the usual sign bootstrapping only extracts one. It therefore corresponds to a slight variation on the original bootstrapping operation and allows us to compute our $k$-NN determination at virtually a constant cost with regard to $k$.

Algorithm 2 presents the modified bootstrapping algorithm that, given an encrypted input, computes the $\texttt{Sign}_1, \ldots, \texttt{Sign}_k$ operations on that input homomorphically. The notations that we use are those found in the original presentation of the bootstrapping algorithm in [Chi+16a] (Algorithm 3: Bootstrapping procedure) and that we reproduced as Algorithm 1. We do this to stress where our algorithm deviates from the original version. Lines 1 to 5 describe the start of the standard bootstrapping procedure. In that original procedure, the last step is an extraction of the first coefficient of the ACC ciphertext. In our variation, we extract $k$ different coefficients to virtually no additional performance cost.

**Remark.** (parallelization)

In section 8.3.1, we mention that the data host is given two bootstrapping keys to compute the homomorphic operations the $k$-NN determination requires. This is because the bootstrapping operation does three things: it outputs a ciphertext with a fixed noise (therefore reducing the input noise if it was higher); it applies a pre-programmed function (here the sign function or its opposite); it switches the encryption key with another one. This key switching means we cannot

**Algorithm 2** Homomorphic `Sign` operations

**Input:** A TLWE sample $(a, b) = [\mu]$ of an unknown message $\mu \in \mathbb{T}$, a bootstrapping key $\mathsf{BK}$, a bootstrapping base $b_\delta$, a number $k$ of desired outputs. The symbol $\boxdot$ represents the internal multiplication operation.

**Output:** $k$ $\mathsf{LWE}$ samples $(u_j)_{1\leq j\leq k}$ where $u_j \in \mathsf{LWE}\left(\frac{1}{b_\delta} \text{ if } \mu + \frac{(j-1)}{b_\delta} \in \left]0, \frac{1}{2}\right[ \ ; \ -\frac{1}{b_\delta} \text{ else}\right)$

1: Let $\overline{b} = \lfloor 2Nb \rceil$ and $\overline{a}_i = \lfloor 2Na_i \rceil$ for each $i \in [1, n]$
2: Let testv $= \left(1 + X + \cdots + X^{N-1}\right) \times X^{-\frac{2N}{4}} \cdot \frac{1}{b_\delta}$
3: $\mathsf{ACC} \leftarrow \left(X^{\overline{b}} \cdot (0, \text{testv})\right)$
4: **for** $i = 1$ **to** $n$
5: $\quad \mathsf{ACC} \leftarrow \left[h + (X^{-\overline{a}_i} - 1) \cdot \mathsf{BK}_i\right] \boxdot \mathsf{ACC}$
6: **for** $j = 1$ **to** $k$
7: $\quad u_j = \mathsf{SampleExtract}\left(\mathsf{ACC}, \ \lfloor \frac{4jN}{b_\delta} \rceil\right)$

re-apply the same bootstrapping operation twice in a row. Therefore Fig. 8.4 does not represent the real order in which the $\delta$ values are used as inputs for the $\mathcal{S}$-boxes.

We can actually design an FHE implementation in the *linear* order presented Fig. 8.4. It would require the use of a key-switching operation after every $\mathcal{S}$-box. This additional operation would decrease the efficiency of the scheme. It would at the same time decrease the bandwidth costs by eliminating the need for a second bootstrapping key (a key-switching key is much lighter than a bootstrapping key).

We presented a *linear* order of computation in Fig. 8.4 for simplicity sake. However, the real precedence relation between the $\mathcal{S}$-boxes offers more degrees of freedom and we were able to implement our scheme with better efficiency by finding of an ordering which removed the need for a key-switching. In essence, we divide our $\delta$ values into chunks of size $m$ and apply $\mathcal{S}$-boxes on each chunk. Then we do the same on the output values of the $\mathcal{S}$-boxes. Additionally, this partial ordering allows for parallelization at run-time, although the timings presented here do not exploit this. Two bootstrapping keys are then needed for the implementation of the scheme.

## 8.4 Experimental Results

We fully implemented the homomorphic $k$-NN scheme described above, and integrated it with a real-world classification problem. In this section we provide the details on our implementation, $k$-NN training and efficiency and accuracy of our FHE $k$-NN classification.

### 8.4.1 FHE Implementation

The choice of parameters depends both on questions of *security* and *accuracy*. The parameters needed to replicate our results are presented in Table 8.2.

As for accuracy, there are two issues that we need to solve to ensure our scheme outputs an accurate result.

One issue is the precision of the bootstrap operation. As shown in Figure 5.2, the bootstrap operation, given a set of parameters, has a red zone around 0 and $\frac{1}{2}$ where input values yield

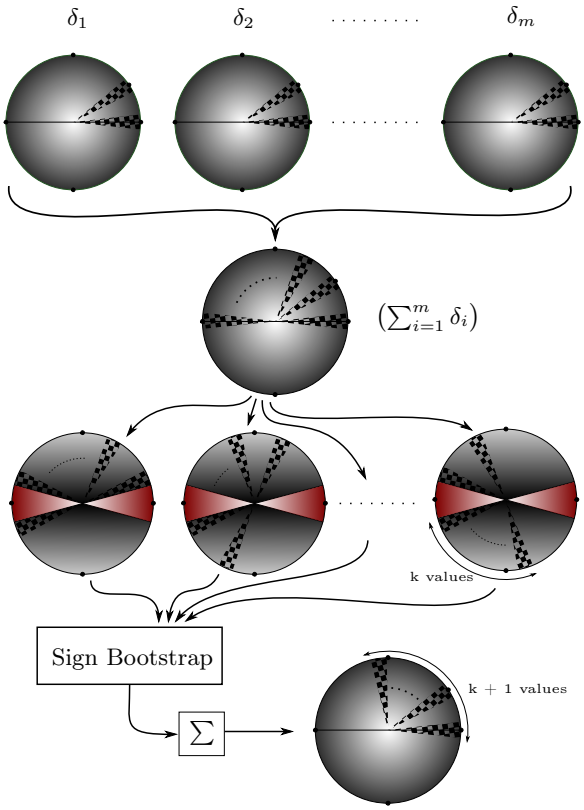


Figure 8.6: A figure showing how the scoring operation intuitively works in the torus. The squared areas in each circle represent every possible value that the given variable can take. For instance, every $\delta$ variable can have either a value of 0 or of $\frac{1}{b_\delta}$. After summing $m$ of them, we obtain a variable that can take one of $m+1$ values.

an uncertain output. We can arrange for the values of $\frac{1}{\nu^2}(\mathbb{d}_i^2 - \mathbb{d}_j^2)$ to never reach the zone around $\frac{1}{2}$ by increasing $\nu$. However, increasing $\nu$ too much will mean having values too small for the bootstrap operation to work accurately. We find the appropriate value for $\nu$ through a scilab script, simulating FHE operations over the dataset on which the homomorphic $k$-NN is implemented (see Section 8.4.2).

As in our argmin scheme presented in Chapter 7, an erroneous output by the bootstrapping operation, when it happens, does not necessarily impact the accuracy of the overall result. If none of the two distances in question is actually one of the $k$ smallest, then the mistake changes nothing to the end-result. In the case where one of the $k$ smallest distances is overlooked and another one is chosen in its stead, we can assume the new distance is close to the one that was overlooked. Therefore in a majority class voting setting (where we determine the class of the source depending on the majority class of its $k$ nearest neighbours) the output determination has a good chance of still being correct. This qualitative analysis does not prove the accuracy of our scheme. Rather it helps explain its theoretical resilience to mistakes. We show in section 8.4.3 that, in practice, our scheme is highly accurate in a real-world setting.

| $\lambda$ | $N$ | $\sigma$ |
|---|---|---|
| 110 | 1024 | 1e−9 |

| $N_b$ | $\sigma_b$ | $B_g$ | $\ell$ |
|---|---|---|---|
| 1024 | 1$e$-9 | 64 | 6 |

| $m$ | $\nu$ | $b_\delta$ | $b_\mathrm{f}$ |
|---|---|---|---|
| 65 | 3 | $4m-4$ | 4 |

Table 8.2: The parameter tables for our implementation. The top-left table presents the overall security ($\lambda$), and the parameters for the initial encryption: $\sigma$ is the Gaussian noise parameter and $N$ the size of polynomials. In the TFHE polynomial encryption scheme (TRLWE), there is a parameter $k$ independent from the one we use to refer to the $k$-NN determination problem here. That parameter was set to 1 in every case. The top-right table presents the parameters needed to create the two bootstrapping keys we are using. The bottom table presents parameters specific to our $k$-NN scheme as presented in Section 8.3. $m$ is the maximum number of sums of fresh $\delta$ ciphertexts we can do before a bootstrapping operation. $\tau$ is used to rescale the data before encryption. $\nu$, $b_\delta$ and $b_\mathrm{f}$ are introduced in Sect. 8.3.5 and are encoding bases used respectively for the initial data, the output of the intermediate scoring operations and the output of the final scoring operation. $b_\delta$ and $b_\mathrm{f}$ are parameters of the scoring operation presented in Algorithm 2.

### 8.4.2 Experimental Setup

#### The datasets

**Breast Cancer dataset.** As a first example of a secure $k$-NN application, we use a dataset[1] previously used with the purpose of testing machine learning algorithms. Specifically, it is a dataset containing 569 instances (vectors) of 30 attributes (meaning the vectors are of dimension $\gamma = 30$). Each of those instances is extracted from a digitized image of a "fine needle aspirate of a breast mass" from a single patient with a tumor in her breast. Every instance is obtained from a different patient. The data is labeled with a class: the tumor can be malignant or benign. This dataset was retrieved from an online open archive of machine learning datasets [DG17]. Our goal is to build a secure $k$-NN classifier that can determine with high accuracy whether a given instance is one of a malignant or a benign tumor by comparing it to an encrypted set of labeled model vectors.

**MNIST dataset.** We present a second application of our algorithm: on the widely used MNIST dataset[2]. This dataset is used to test the classification of handwritten digits (0 to 9). In fact, we do not use the raw MNIST dataset (which consists of 60,000 training images and 10,000 testing images all of them with 28x28 pixels) but we use a pre-processed subset of the data provided by the scikit-learn[3] library. This subset includes 1,797 images of size 8x8.

#### The classification process

**About the classification process.** In order to showcase the efficiency and accuracy of our scheme, we design our own $k$-NN classification algorithm over the dataset. Since, as discussed in Sect. 8.2.3, we are interested only in the homomorphic evaluation of *already trained* $k$-NN classifiers, we train our algorithm in the clear domain. In this case, *training* means selecting a subset of the learning data that will be used as reference vectors (model vectors) in a $k$-NN computation as well as selecting the most appropriate $k$ value. The classification is then

[1] `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`
[2] http://yann.lecun.com/exdb/mnist/
[3] https://scikit-learn.org/stable/

made homomorphically and consists of running our $k$-NN algorithm over the encrypted reference vectors selected in the training phase, comparing them to a cleartext instance. The results are then decrypted and a majority voting takes place: the instance is attributed the class shared by a majority of the $k$ model vectors selected as its closest.

With respect to this, our goal is not to design the best clear-domain classification algorithm for the given problem but rather to build a "good enough" clear-domain algorithm and show that its encrypted-domain counterpart works just as well or almost as well and efficiently so. In particular, with respect to efficiency, as the computing time of any algorithm running in the encrypted domain over FHE *cannot by construction depend on its input data*, we could even have used a random selection of reference vectors and obtained the very same performance results.

**Breast Cancer dataset.** The data is first pre-processed by rescaling every attribute to a value between 0 and 1. More complex classification algorithms could choose to rescale some attributes differently to increase or decrease their weight in the distance computation. However, most of the previous work applying a $k$-NN algorithm applies this simple pre-processing (see [Ska+99]). We simply apply the same pre-processing. As seen section 8.3.5 our model vector and source vector values are both rescaled using the same factor $\nu$ and the source vector is multiplied by a factor $\tau$ and then rounded.

At this point a classification rate is obtained through a classification process which we took from the existing literature (see [Ska+99] for instance). A few things change in our process compared with the literature's accepted standard. We present our reasoning below. Here is our classification process:

- 1. Select 459 instances at the start to be the testing set among the 569.
- 2. Among the 110 instances left, select randomly 10 instances to be the training set. The 100 other one are the validation set.
- 3. Classify the 100 instances from the validation set by finding their $k$ closest neighbors among the training set instances with $k$ an odd number. The majority class among the neighbors is selected.
- 4. Compute the classification rate (the number of correct classifications divided by 100).
- 5. Repeat steps 2 to 4 a certain amount of times and keep the training set that yields the maximum classification rate on the validation set.
- 6. Compute the classification rate of the testing set using the selected best training set. This is what we actually call the classification rate.

The use of a training, validation and testing set is a standard machine learning method. In this context, one aims to optimize one's classification rate on the validation data by changing every parameter under their control (here, the size and content of the training set and the value $k$). Then the classification rate of the testing set is computed once.

We mention that we could have chosen to obtain better classification rates by selecting a larger training set and a smaller testing set. A rule of thumb in machine learning is to have the following ratio: 64% training set; 16% validation set; 20% testing set. This ratio is not the one we choose to apply: a very small (in absolute value) testing set could not have yielded a very trustworthy result. In comparison, our ratio is the following: 2% training set; 18% validation set; 80% testing set. With this ratio, we achieve classification rates comparable to the state-of-the-art results

on this database using the $k$-NN classification method. It allows us to reduce the time a single classification takes in the encrypted space, and, by increasing the size of the testing set, to make our results more stable.

For reproducibility purposes, we include here the identification numbers of all 10 vectors eventually selected as our model vectors: 864018, 857010, 855563, 848406, 857156, 862717, 8510653, 86208, 861598, 852781.

**MNIST dataset.** The database in this case is pre-processed and therefore we jump directly to the selection of the model vectors phase. We do this in exactly the same manner as with the breast cancer database. Importantly, we used several different sizes for our model to give a better understanding of our algorithm's scaling ability, both in terms of accuracy and in terms of efficiency.

### 8.4.3 Performance Results

#### Classification rate

**Breast Cancer dataset.** For a measure of our algorithm's accuracy, we can compare with the results of previous work (see [Ska+99] for instance) which regularly present classification rates between 94% and 97% depending on the method used. We train our $k$-NN classifier in the clear and obtain a classification rate of 95.2% with $k = 3$ and $d = 10$ model vectors.

We implemented the FHE classification algorithm and ran it on the encrypted model vectors selected during the training phase. The overall classification rate was 95.0%. This means "going FHE" only slightly reduces the accuracy of our $k$-NN classifier. Therefore, rounding the classified instances to integer vectors and keeping real model vectors does not affect the overall accuracy that much. Furthermore, our algorithm works well here not because all of the instances are quite far from one another but rather in spite of the opposite. With the given parameters, we determined that around 4% of the differences of squared distances are lower than the precision threshold of our sign bootstrapping operation. However, as mentioned in Section 8.4.1, a $k$-NN classifier can be quite resilient to such computational mistakes.

In all of our testing, we found values for $k$ that are low (7 or below) not because it was easier for us computationally (we mentioned that it changes almost nothing to the efficiency of the algorithm) but rather because $k$ values are always very low in a $k$-NN classifier. [Fen+93] reviewed an important amount of $k$-NN algorithms at the time and determined that in 75% of the cases, the $k = 1$ classifier was the best one.

Interestingly, in the FHE case, we found that $k = 3$ gives significantly better results than $k = 1$. When $k = 1$, the difference in classification rate between a clear classification and an FHE one is 3% on average. The difference when $k = 3$ never exceeds 0.4%. This is due to the fact that, as mentioned in Sect. 8.4.1, a majority-class voting scenario ($k > 1$) attenuates the impact of a single error significantly.

**MNIST dataset.** An application of our algorithm on the well-known MNIST database allows us to provide benchmarks for the efficiency and accuracy of our algorithm. To that end, we test our algorithm with 5 different model sizes. Each size is chosen as the smallest model size able to achieve a given benchmark classification rate on the MNIST database. Table 8.3 shows the values that were chosen for model sizes ($d$) and their respective benchmark clear classification

rates. It shows the classification rates achieved with each model through the FHE classifier. We use $k = 3$ for every classification.

| $d$ | 40 | 175 | 228 | 269 | 457 |
|---|---|---|---|---|---|
| clear rate (%) | 80 | 95 | 96 | 97 | 98 |
| fhe rate (%) | 79.5 | 94.8 | 95.4 | 96.4 | 97.3 |

Table 8.3: k-NN classification rates over the MNIST database. This table gives the MNIST classification rates both in the clear case and in the fhe case, depending on the size of the model ($d$).

Overall the accuracy loss is relatively low. Importantly, it does not depend on the size of the model: this is due to the fact that we built a fully homomorphic scheme and not a levelled scheme. Since the parameters are the same regardless of the size of the problem, on average, the loss of accuracy is constant. The relatively high number of model vectors needed to obtain a similar classification rate to that of the breast cancer database result can be explained (in part) by the higher number of classes. The MNIST database requires a classification among 10 different classes (there are 2 classes for the breast cancer database), this increases the amount of model vectors needed for an accurate classification.

### 8.4.4 Timing Results

As mentioned previously, our algorithm has a quadratic complexity with regard to the number of training instances (model vectors). This is because we are computing $\frac{d^2-d}{2}$ sign bootstrapping operations to obtain the $\delta$ values. After that, we need to run the scoring algorithm on each column of the $\delta$ matrix, with roughly $d \times \frac{d}{m-k}$ $\mathcal{S}$-box evaluations in total. Of course, this is a less efficient $k$-NN algorithm compared with existing work in the clear as, some optimizations in the literature allow *approximate* $k$-NN classifiers to run in sub-linear time. However our work is the only one to achieve a fully non-interactive secure $k$-NN classifier, and the quadratic complexity is the price that we pay for that. This can be explained intuitively by the fact that any such secure non-interactive classifier cannot use conditional gates to reduce the complexity of the algorithm: if it did, it would leak a lot of information to the data host. We are running the operations on an Intel Core i7-6600U CPU. All of the times that we present here correspond to sequential computation times. Our scheme is highly parallelizable as mentioned.

**Breast Cancer dataset.** The real-world breast cancer detection problem we applied our algorithm on requires us to find the $k = 3$ closest vectors to a source vector among $d = 10$ model vectors. When applied to this problem, a single $k$-NN classification takes 4 seconds to finish *sequentially*. In the first application scenario presented in Sect. 8.2.3, a patient and/or her doctor are sending an encrypted query for remote classification. In our case this means they would have to wait around 4 seconds for an answer. Given that an average medical consultation takes around 15 minutes, the response time is appropriate. In the second scenario of an epidemiological study as presented in Sect. 8.2.3, a pharmaceutical company running this study over several remote databases of 500 patient vectors *each* would have to wait under 35 minutes (for *sequential* computations) to get the results of their study. It is an appropriate latency for such a use-case.

**MNIST dataset.** As for the MNIST database, Table 8.4 shows how much time a single classification takes depending on the size of the model. For instance, in order to classify with 95% accuracy, one classification would take just under 12 minutes of *sequential* computation-time to complete (recall that our scheme is highly parallelizable and that these timings can be significantly decreased through multi-core execution).

| $d$ | 40 | 175 | 228 | 269 | 457 |
|---|---|---|---|---|---|
| time (min) | 0.5 | 11.6 | 18.3 | 25.4 | 70.8 |

Table 8.4: Sequential timings for a single MNIST k-NN homomorphic classification, depending on the size of the model ($d$). The time is given in minutes.

### 8.4.5 Bandwidth usage

In the $k$-NN scenario presented in Sect. 8.3.5, the model vectors are encrypted and the query is in the clear. For the breast-cancer problem at hand, it means 10 $\left[C^{(i)}\right]^{(r)}$ ciphertexts and 10 $[A_i]^{(r)}$ ciphertexts. Given the parameters chosen for the implementation (see Sect. 8.4.1), the encrypted data to be transferred amounts to 80 KBytes. Additionally, the two bootstrapping keys which are transferred once and for all at setup time, take 200 MBytes.
In the case where our scheme is used to send an encrypted query to be compared with a set of clear model vectors, then the communication overhead prior to the computation amounts to a single TRLWE sample (8 KBytes) and two bootstrapping keys (200 MBytes). Again, these keys are transferred once, at setup time.
The encrypted result is sent in every case as 10 TLWE samples amounting to 40 KBytes.

## 8.5 Conclusion

We are not the first to present a confidentiality-preserving $k$-NN classifier, however, to the best of our knowledge, we are the only ones to produce a fully homomorphic, non-interactive algorithm. While our scheme has obvious drawbacks - namely its quadratic complexity - we believe that it can be used in a real world setting already, and can be made more efficient with future improvements (parallelization is one of them). There is a significant increase in accuracy when using our $k$-NN FHE classifier compared with our 1-NN classifier. This confirms the relevance of our work as an addition to our previous result on an argmin operator (Chapter 7).

# Chapter 9

# Conclusion

## 9.1 The challenges of confidential machine learning

The last ten years have seen a small revolution in the world of homomorphic encryption. From an abstract challenge with no theoretical solution in sight, the field evolved to produce its first practical solutions. FHE - especially general purpose FHE - is still a costly addition to any application in terms of efficiency. General purpose machine learning algorithms provide a unique opportunity to produce special purpose FHE tools for confidentiality-preserving alternatives. Among the most important decisions we had to make during our research was that of choosing which of these algorithms we could adapt into FHE alternatives.

When starting a PhD on the subject of homomorphic encryption it seems that one of the catchiest titles could be "Practical homomorphic encryption for Deep Convolutional Neural Networks". Machine learning is one of the most useful and popular technological tools being developed today. The rise of "artificial intelligence" has fascinated computer scientists and science-fiction writers alike for decades. Its promises and dangers are creating new moral dilemmas and reshaping our understanding of life itself. Among the machine learning algorithms at the forefront of this technological revolution are deep CNNs. The question then is why this thesis is *not* called "Practical homomorphic encryption for Deep Convolutional Neural Networks". Indeed, if this paragraph is to achieve anything in the mind of the reader, it should be to convince them that this line of research is more than worthy.

However, our belief from the beginning has been that an intense research focus on feed-forward neural networks should not prevent us to look for practical confidentiality-preserving applications elsewhere. Actually, too intense a focus on feed-forward neural networks might even prevent the research community to find hidden gems of secure machine learning where no one could have expected them. Deviating from the main road is the best way to get lost, but also the only way to find uncharted oases of confidentiality-preserving machine learning goodness.

From this belief emerged the idea to build an homomorphic discrete recursive neural network, a model-confidential embedding-based neural network, and the first non-interactive confidentiality-preserving $k$-NN classifier. Along the way, we have considered other possible applications that we either could not find a way to adapt for FHE, or are promising perspectives on which we are eager to build our future work. Perspectives aside for now, let us review our work summarily.

## 9.2 Our contributions

The choice of Hopfield networks was made as an answer to the question: how can we find FHE-friendly algorithms on which to build an efficient, secure classification ? In retrospect, the answer should have been to look for simple machine learning algorithms that are widely used today, and not a rather obscure recursive neural network. For this reason, we do not think Hopfield networks will see an increase in popularity in the wake of our work. However, we do hope that the spirit with which we set out to develop our homomorphic Hopfield network proves useful for future research: the search for simple FHE-friendly machine learning algorithms.

In our second contribution, we looked at a best-of-class, modern and efficient classifier. What showed promise was the separation that the VGGVox model had between a complex set of CNNs and the classification model that they are trained to create. Our goal was to produce a version of their classifier where the model could be protected non-interactively during the classification phase. This second work aimed at adapting a modern and highly efficient algorithm with a set of restricted confidentiality-preserving goals.

Our third work aimed at providing an efficient, practical confidentiality-preserving alternative to one of the bedrocks of machine learning: the $k$-NN classifier. In that respect, it is the work that promises the most impact. Thanks to the versatility of the $k$-NN classifier, we see our homomorphic alternative as a a good FHE building block for other secure machine learning algorithms.

## 9.3 Perspectives for future work

Where to go from here ? There are a lot of perspectives for secure machine learning using FHE in general. Here, we want to focus on exactly what research possibilities *our* work opens us up to. The main unresolved question throughout this thesis is the perspective of a secure training phase for complex algorithms. We recall that one of our motivations for building an argmax operator was the perspective of using it in a larger training algorithm. So why didn't we do it ?

**Naive homomorphic training.** If one were to imagine using a novel homomorphic training algorithm over a feed-forward neural network, how would one fare ? Let's take the case of the homomorphic neural network built in [Bou+18] and assume we are trying to train it homomorphically in the most naive way possible. Their neural network takes about 1.6s to evaluate homomorphically (their 100 hidden neurons version). If we were to imagine training it homomorphically, we would first have to determine a way to compute an approximate gradient descent on encrypted data. Assuming we can do it and one gradient descent takes as much time to compute as an evaluation (i.e. 1.6s), then we can run the training algorithm. The MNIST dataset (used in [Bou+18]) has 60000 training images, if every one of them is used 100 times during the training algorithm (a low number according to the literature), then we can expect the training process to take a little over 222 days to complete.

This is obviously ludicrous and we are only taking the most naive approach to underscore the need to adapt the existing training algorithms or try to address different - if similar - challenges more fitted for FHE adaptation.

**Secure federated learning.** One such challenge consists of addressing the confidentiality matters surrounding the use of *federated learning*. Traditional training uses two sets of data

(training and testing) on a single server running the training algorithm. The problem is the following: data is hard to come by, and good - labeled - data even harder.

Federated learning allows multiple users - teachers - to train a single machine learning model together without directly sending their labeled training data to a centralized server. Instead, they first train their own models - called teacher models - with their respective datasets. Then, using their teacher model, every user classifies data from an unlabeled public dataset and sends it to the server. It then chooses the correct label for each data point by applying a majority vote among all of the teacher's answers. This creates a labeled dataset which is used to train the student model.

Ideally, this new model was obtained without any information about the original training datasets leaking to the server. In practice, the information that every teacher sends to the server is sensitive and needs to be protected. This can be achieved by encrypting the data sent by the teachers and using differential privacy and our homomorphic argmax operator for the homomorphic majority vote the server needs to run. For details about how this works in practice see [Séb+20], an article under submission on which we collaborated to provide the homomorphic majority vote.

**What else ?** While federated learning offers a great opportunity to achieve confidentiality goals over decentralized learning, it is not the only training scheme that can make use of our argmin/argmax operator. Indeed, to our knowledge, there is no training algorithm that does not require some sort of selection, and therefore could be a good match for homomorphic adaptation. Of course, it is not that simple. Most training algorithms are complex. In general, the "better" a machine learning algorithm is, the more complex its training algorithm. The example of federated learning - which can be applied on the most complex neural networks existing today - shows that a lot can be achieved when not considering a straightforward non-interactive homomorphic application of a stochastic gradient descent for a deep convolutional neural network.

Other perspectives are aplenty: lowering the complexity of our $k$-NN operator; implementing an efficient MUX gate using the strengths of the Chimera framework (arguably a good step toward that first goal); have a look at secure DNA-matching techniques (although it was our initial example for the need for confidentiality, we never addressed the matter in our contributions) ...

# Bibliography


[Abd+20] Michel Abdalla, Florian Bourse, Hugo Marival, David Pointcheval, Azam Soleimanian, and Hendrik Waldner. "Multi-Client Inner-Product Functional Encryption in the Random-Oracle Model". In: *Security and Cryptography for Networks*. Ed. by Clemente Galdi and Vladimir Kolesnikov. Cham: Springer International Publishing, 2020, pp. 525–545. ISBN: 978-3-030-57990-6.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. "Faster Bootstrapping with Polynomial Error". In: *Advances in Cryptology – CRYPTO 2014*. Ed. by Juan A. Garay and Rosario Gennaro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 297–314. ISBN: 978-3-662-44371-2.

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

[AS00] Rakesh Agrawal and Ramakrishnan Srikant. "Privacy-Preserving Data Mining". In: SIGMOD '00 (2000), pp. 439–450. DOI: 10.1145/342009.335438. URL: https://doi.org/10.1145/342009.335438.

[Ash+18] Gilad Asharov, Shai Halevi, Yehuda Lindell, and Tal Rabin. "Privacy-Preserving Search of Similar Patients in Genomic Data". In: *Proceedings on Privacy Enhancing Technologies* 2018 (Oct. 2018), pp. 104–124. DOI: 10.1515/popets-2018-0034.

[Bal+19] Marshall Ball, Brent Carmer, Tal Malkin, Mike Rosulek, and Nichole Schimanski. *Garbled Neural Networks are Practical*. Cryptology ePrint Archive, Report 2019/338. 2019.

[BD10] M. Burkhart and X. Dimitropoulos. "Fast Privacy-Preserving Top-k Queries Using Secret Sharing". In: (Aug. 2010), pp. 1–7.

[BEG19] Alon Brutzkus, Oren Elisha, and Ran Gilad-Bachrach. "Low Latency Privacy Preserving Inference". In: *International Conference on Machine Learning*. 2019.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ITCS '12. Cambridge, Massachusetts: Association for Computing Machinery, 2012, pp. 309–325. ISBN: 9781450311151. DOI: 10.1145/2090236.2090262. URL: https://doi.org/10.1145/2090236.2090262.

[Bin+18] Nina Bindel, Johannes Buchmann, Florian Göpfert, and Markus Schmidt. "Estimation of the hardness of the learning with errors problem with a restricted number of samples". In: *Journal of Mathematical Cryptology* 13 (Aug. 2018). DOI: 10.1515/jmc-2017-0040.

[Boe+19] Fabian Boemer, Yixing Lao, Rosario Cammarota, and Casimir Wierzynski. "NGraph-HE: A Graph Compiler for Deep Learning on Homomorphically Encrypted Data". In: *Proceedings of the 16th ACM International Conference on Computing Frontiers*. CF '19. Alghero, Italy: Association for Computing Machinery, 2019, pp. 3–13. ISBN:

9781450366854. DOI: 10.1145/3310273.3323047. URL: https://doi.org/10.1145/3310273.3323047.

[Bon+04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. "Public Key Encryption with Keyword Search". In: *Advances in Cryptology - EUROCRYPT 2004*. Ed. by Christian Cachin and Jan L. Camenisch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 506–522. ISBN: 978-3-540-24676-3.

[Bou+16] Florian Bourse, Rafaël Del Pino, Michele Minelli, and Hoeteck Wee. "FHE Circuit Privacy Almost for Free". In: *Advances in Cryptology – CRYPTO 2016*. Ed. by Matthew Robshaw and Jonathan Katz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 62–89. ISBN: 978-3-662-53008-5.

[Bou+18] F. Bourse, M. Minelli, M. Minihold, and P. Paillier. "Fast Homomorphic Evaluation of Deep Discretized Neural Networks". In: (2018).

[Bou+20] Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. "CHIMERA: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes". In: *Journal of Mathematical Cryptology* 14.1 (1Jan. 2020), pp. 316–338. DOI: https://doi.org/10.1515/jmc-2019-0026. URL: https://www.degruyter.com/view/journals/jmc/14/1/article-p316.xml.

[Bra+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical Hardness of Learning with Errors". In: *Proceedings of the Annual ACM Symposium on Theory of Computing* (June 2013). DOI: 10.1145/2488608.2488680.

[Bra12] Zvika Brakerski. "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP". In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 868–886. ISBN: 978-3-642-32009-5.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. "Functional Encryption: Definitions and Challenges". In: *Theory of Cryptography*. Ed. by Yuval Ishai. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 253–273. ISBN: 978-3-642-19571-6.

[BV11a] Z. Brakerski and V. Vaikuntanathan. "Efficient Fully Homomorphic Encryption from (Standard) LWE". In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. 2011, pp. 97–106.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages". In: *Advances in Cryptology – CRYPTO 2011*. Ed. by Phillip Rogaway. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 505–524. ISBN: 978-3-642-22792-9.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. "Lattice-based FHE as secure as PKE". In: Jan. 2014, pp. 1–12. DOI: 10.1145/2554797.2554799.

[Can+17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. "Chosen-Ciphertext Secure Fully Homomorphic Encryption". In: *Public-Key Cryptography – PKC 2017*. Ed. by Serge Fehr. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 213–240. ISBN: 978-3-662-54388-7.

[CDS15] Sergiu Carpov, Paul Dubrulle, and Renaud Sirdey. "Armadillo: a compilation chain for privacy preserving applications". In: *Proceedings of the 3rd International Workshop on Security in Cloud Computing*. 2015, pp. 13–19.

[Cha+17] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. *Privacy-Preserving Classification on Deep Neural Network*. Cryptology ePrint Archive, Report 2017/035. 2017.

[Cha+19] Hervé Chabanne, Roch Lescuyer, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. "Recognition Over Encrypted Faces". In: *Mobile, Secure, and Programmable Networking*. Ed. by Éric Renault, Selma Boumerdassi, and Samia Bouze-

frane. Cham: Springer International Publishing, 2019, pp. 174–191. ISBN: 978-3-030-03101-5.

[Che+17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic Encryption for Arithmetic of Approximate Numbers". In: (2017). Ed. by Tsuyoshi Takagi and Thomas Peyrin.

[Che+19] Hao Chen, Ilaria Chillotti, Yihe Dong, Oxana Poburinnaya, Ilya P. Razenshteyn, and M. Sadegh Riazi. "SANNS: Scaling Up Secure Approximate k-Nearest Neighbors Search". In: *CoRR* (2019).

[Chi+16a] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. "Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds". In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 3–33. ISBN: 978-3-662-53887-6.

[Chi+16b] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. *TFHE: Fast Fully Homomorphic Encryption Library*. https://tfhe.github.io/tfhe/. August 2016.

[Chi+17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. "Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE". In: *ASIACRYPT*. 2017.

[Chi+19] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. "TFHE: Fast Fully Homomorphic Encryption Over the Torus". In: *Journal of Cryptology* 33 (Apr. 2019). DOI: 10.1007/s00145-019-09319-x.

[CKL15] Jung Hee Cheon, Miran Kim, and Kristin Lauter. "Homomorphic Computation of Edit Distance". In: *Financial Cryptography and Data Security*. Ed. by Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 194–212. ISBN: 978-3-662-48051-9.

[CLP17] Hao Chen, Kim Laine, and Rachel Player. "Simple Encrypted Arithmetic Library - SEAL v2.1". In: (2017).

[CNZ18a] J. S. Chung, A. Nagrani, and A. Zisserman. "VoxCeleb2: Deep Speaker Recognition". In: (2018).

[CNZ18b] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. "VoxCeleb2: Deep Speaker Recognition". In: *CoRR* (2018).

[CS98] Ronald Cramer and Victor Shoup. "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack". In: *Advances in Cryptology — CRYPTO '98*. Ed. by Hugo Krawczyk. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 13–25. ISBN: 978-3-540-68462-6.

[DG17] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[DH06] W. Diffie and M. Hellman. "New Directions in Cryptography". In: *IEEE Trans. Inf. Theor.* 22.6 (Sept. 2006), pp. 644–654. ISSN: 0018-9448. DOI: 10.1109/TIT.1976.1055638. URL: https://doi.org/10.1109/TIT.1976.1055638.

[DL20] Emile Dirks and James Leibold. *Genomics surveillance, inside China's DNA dragnet*. Tech. rep. Australian Strategic Policy Institute, 2020.

[DM15] Léo Ducas and Daniele Micciancio. "FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second". In: *Advances in Cryptology – EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 617–640. ISBN: 978-3-662-46800-5.

[Dow+16] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. "CryptoNets: Applying Neural Networks to Encrypted Data

with High Throughput and Accuracy". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 201–210.

[DP20] Damien Desfontaines and Balázs Pejó. "SoK: Differential privacies". In: *Proceedings on Privacy Enhancing Technologies* 2020.2 (2020), pp. 288–313. DOI: https://doi.org/10.2478/popets-2020-0028. URL: https://content.sciendo.com/view/journals/popets/2020/2/article-p288.xml.

[DR14] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Found. Trends Theor. Comput. Sci.* 9.3–4 (Aug. 2014), pp. 211–407. ISSN: 1551-305X. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

[DS16] Léo Ducas and Damien Stehlé. "Sanitization of FHE Ciphertexts". In: *Advances in Cryptology – EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 294–310. ISBN: 978-3-662-49890-3.

[DSB17] Denise Demirel, Lucas Schabhüser, and Johannes A. Buchmann. "Privately and Publicly Verifiable Computing Techniques - A Survey". In: *Springer Briefs in Computer Science*. 2017.

[Dwo06] Cynthia Dwork. "Differential Privacy". In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12. ISBN: 978-3-540-35908-1.

[EJK20] Thomas Espitau, Antoine Joux, and Natalia Kharchenko. *On a hybrid approach to solve small secret LWE*. Cryptology ePrint Archive, Report 2020/515. https://eprint.iacr.org/2020/515. 2020.

[Fen+93] C Feng, A Sutherland, R King, S Muggleton, and R Henery. "Comparison of machine learning classifiers to statistics and neural networks". In: (1993), pp. 41–52.

[FGP14] Dario Fiore, Rosario Gennaro, and Valerio Pastro. "Efficiently Verifiable Computation on Encrypted Data". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. CCS '14. Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, pp. 844–855. ISBN: 9781450329576. DOI: 10.1145/2660267.2660366. URL: https://doi.org/10.1145/2660267.2660366.

[FNP20] Dario Fiore, Anca Nitulescu, and David Pointcheval. "Boosting Verifiable Computation on Encrypted Data". In: *Public-Key Cryptography – PKC 2020*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Cham: Springer International Publishing, 2020, pp. 124–154. ISBN: 978-3-030-45388-6.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[Gen09a] Craig Gentry. "A Fully Homomorphic Encryption Scheme". PhD thesis. Stanford, CA, USA, 2009. ISBN: 9781109444506.

[Gen09b] Craig Gentry. "Fully Homomorphic Encryption Using Ideal Lattices". In: STOC '09 (2009).

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. "Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers". In: *Advances in Cryptology – CRYPTO 2010*. Ed. by Tal Rabin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 465–482. ISBN: 978-3-642-14623-7.

[GH11] Craig Gentry and Shai Halevi. "Implementing Gentry's Fully-Homomorphic Encryption Scheme". In: *Advances in Cryptology – EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 129–148. ISBN: 978-3-642-20465-4.

[GM82] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information". In: *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*. STOC '82. San Francisco, California, USA: Association for Computing Machinery, 1982, pp. 365–377. ISBN: 0897910702. DOI: 10.1145/800070.802212. URL: https://doi.org/10.1145/800070.802212.

[GM84] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption". In: *Journal of Computer and System Sciences* 28.2 (1984), pp. 270–299. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(84)90070-9. URL: http://www.sciencedirect.com/science/article/pii/0022000084900709.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based". In: (2013).

[Gur97] Kevin Gurney. "An introduction to neural networks". In: (1997).

[Hop82] J J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pp. 2554–2558. ISSN: 0027-8424. DOI: 10.1073/pnas.79.8.2554. eprint: http://www.pnas.org/content/79/8/2554.full.pdf. URL: http://www.pnas.org/content/79/8/2554.

[HSD11] Jonathon S. Hare, Sina Samangooei, and David P. Dupplaw. "OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images". In: (2011).

[Hu+11] H. Hu, J. Xu, C. Ren, and B. Choi. "Processing private queries over untrusted data cloud through privacy homomorphism". In: (Apr. 2011), pp. 601–612.

[ISZ19] Malika Izabachène, Renaud Sirdey, and Martin Zuber. "Practical Fully Homomorphic Encryption for Fully Masked Neural Networks". In: *Cryptology and Network Security*. Ed. by Yi Mu, Robert H. Deng, and Xinyi Huang. Cham: Springer International Publishing, 2019, pp. 24–36. ISBN: 978-3-030-31578-8.

[JF12] J. Fan and F. Vercauteren. "Somewhat practical fully homomorphic encryption". 2012.

[KC04] Murat Kantarcıoğlu and Chris Clifton. "Privately Computing a Distributed k-nn Classifier". In: (2004). Ed. by Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, pp. 279–290.

[KMC18] Julien Keuffer, Refik Molva, and Hervé Chabanne. "Efficient Proof Composition for Verifiable Computation". In: *Computer Security*. Ed. by Javier Lopez, Jianying Zhou, and Miguel Soriano. Cham: Springer International Publishing, 2018, pp. 152–171. ISBN: 978-3-319-99073-6.

[LLV07] N. Li, T. Li, and S. Venkatasubramanian. "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity". In: *2007 IEEE 23rd International Conference on Data Engineering*. 2007, pp. 106–115.

[LP00] Yehuda Lindell and Benny Pinkas. "Privacy Preserving Data Mining". In: (2000). Ed. by Mihir Bellare, pp. 36–54.

[LPR10] V Lyubashevsky, C. Peikert, and O. Regev. "On ideal lattices and learning with errors over rings". In: (2010).

[LPR12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *On Ideal Lattices and Learning with Errors Over Rings*. Cryptology ePrint Archive, Report 2012/230. 2012.

[LSP15] Frank Li, Richard Shin, and Vern Paxson. "Exploring Privacy Preservation in Outsourced K-Nearest Neighbors with Multiple Data Owners". In: (July 2015).

[Mac+06] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. "L-diversity: privacy beyond k-anonymity". In: *22nd International Conference on Data Engineering (ICDE'06)*. 2006, pp. 24–24.

[Mac03] David MacKay. "Information Theory, Inference, and Learning Algorithms". In: (2003).

[Mar+19] Tilen Marc, Miha Stopar, Jan Hartman, Manca Bizjak, and Jolanda Modic. *Privacy-Enhanced Machine Learning with Functional Encryption*. Ed. by Kazue Sako, Steve Schneider, and Peter Y. A. Ryan. Cham, 2019.

[MGH10] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. "Additively Homomorphic Encryption with d-Operand Multiplications". In: *Advances in Cryptology – CRYPTO 2010*. Springer Berlin Heidelberg, 2010, pp. 138–154. DOI: 10.1007/978-3-642-14623-7_8. URL: https://doi.org/10.1007%2F978-3-642-14623-7_8.

[MS20] O. Stan M. Abbass and R. Sirdey. "Computing neural networks with homomorphic encryption and verifiable computing". In: *Proceedings of the 2nd Workshop on Cloud Security and Privacy 2020*. (to appear). 2020.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. 1st. USA: CRC Press, Inc., 1996. ISBN: 0849385237.

[NCZ17] A. Nagrani, J. S. Chung, and A. Zisserman. "VoxCeleb: a large-scale speaker identification dataset". In: (2017).

[Pai99] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *Advances in Cryptology — EUROCRYPT '99*. Ed. by Jacques Stern. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238. ISBN: 978-3-540-48910-8.

[PL18] Jeongsu Park and Dong Lee. "Privacy Preserving k -Nearest Neighbor for Medical Diagnosis in e-Health Cloud". In: *Journal of Healthcare Engineering* 2018 (Oct. 2018), pp. 1–11.

[Pla18] Rachel Player. "Parameter selection in lattice-based cryptography". PhD thesis. University of London, Oct. 2018.

[QA08] Y. Qi and M. J. Atallah. "Efficient Privacy-Preserving k-Nearest Neighbor Search". In: (June 2008), pp. 311–319.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. "On Data Banks and Privacy Homomorphisms". In: *Foundations of Secure Computation, Academia Press* (1978), pp. 169–179.

[Reg05] Oded Regev. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: (2005).

[Ron+16] H. Rong, H. Wang, J. Liu, and M. Xian. "Privacy-Preserving k-Nearest Neighbor Computation in Multiple Cloud Environments". In: *IEEE Access* 4 (2016), pp. 9589–9603.

[RRK17] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. "DeepSecure: Scalable Provably-Secure Deep Learning". In: *CoRR* (2017).

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

[Ryf+19] Théo Ryffel, Edouard Dufour-Sans, Romain Gay, Francis Bach, and David Pointcheval. "Partially Encrypted Machine Learning using Functional Encryption". In: *NeurIPS 2019 - Thirty-third Conference on Neural Information Processing Systems*. Advances in Neural Information Processing Systems. Vancouver, Canada, Dec. 2019. URL: https://hal.inria.fr/hal-02357181.

[Séb+20] Arnaud Grivet Sébert, Rafael Pinot, Martin Zuber, Cédric Gouy-Pailler, and Renaud Sirdey. "SPEED: Secure, PrivatE, and Efficient Deep learning". In: *CoRR* abs/2006.09475 (2020). (submitted for publication).

[SEJ15] B. K. Samanthula, Y. Elmehdwi, and W. Jiang. "k-Nearest Neighbor Classification over Semantically Secure Encrypted Relational Data". In: *IEEE Transactions on Knowledge and Data Engineering* 27.5 (2015), pp. 1261–1273.

[Sha84] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*. Vol. 196. Lecture Notes in Computer Science. Springer, 1984, pp. 47–53. DOI: 10.1007/3-540-39568-7_5.

[Ska+99] David Skalak, Paul Utgoff, Andrew Barto, Michael Member, and David Stemple. "Prototype Selection For Composite Nearest Neighbor Classifiers". In: (July 1999).

[SKK06] M. Shaneck, Y. Kim, and V. Kumar. "Privacy Preserving Nearest Neighbor Search". In: (Dec. 2006), pp. 541–545.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *CoRR* (2015).

[SS98] P. Samarati and L. Sweeney. *Protecting Privacy when Disclosing Information: k-Anonymity and its Enforcement through Generalization and Suppression*. Tech. rep. 1998.

[ST12] Klara Stokes and Vicenç Torra. "N-confusion: A generalization of k-anonymity". In: *ACM International Conference Proceeding Series* (Mar. 2012). DOI: 10.1145/2320765.2320824.

[SV10] N. P. Smart and F. Vercauteren. "Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes". In: *Public Key Cryptography – PKC 2010*. Ed. by Phong Q. Nguyen and David Pointcheval. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 420–443. ISBN: 978-3-642-13013-7.

[SW05] Amit Sahai and Brent Waters. "Fuzzy Identity-Based Encryption". In: *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*. EUROCRYPT'05. Aarhus, Denmark: Springer-Verlag, 2005, pp. 457–473. ISBN: 3540259104. DOI: 10.1007/11426639_27. URL: https://doi.org/10.1007/11426639_27.

[SZ07] Humayun Karim Sulehria and Ye Zhang. "Hopfield Neural Networks: A Survey". In: (2007).

[TT10] Tan, Xiaoyang and Triggs, Bill. "Enhanced local texture feature sets for face recognition under difficult lighting conditions". In: *Trans. Img. Proc.* 19 (2010).

[VJ10] Marten Van Dijk and Ari Juels. "On the Impossibility of Cryptography Alone for Privacy-preserving Cloud Computing". In: HotSec'10 (2010), pp. 1–8.

[Won+09] Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. "Secure kNN Computation on Encrypted Databases". In: SIGMOD '09 (2009), pp. 139–152.

[XCL06] Li Xiong, Subramanyam Chitti, and Ling Liu. "K Nearest Neighbor Classification across Multiple Private Databases". In: (2006).

[XCL07] Li Xiong, Subramanyam Chitti, and Ling Liu. "Preserving data privacy in outsourcing data aggregation services". In: *ACM Trans. Internet Techn.* 7 (Aug. 2007).

[Xie+14] Pengtao Xie, Misha Bilenko, Tom Finley, Ran Gilad-Bachrach, Kristin E. Lauter, and Michael Naehrig. "Crypto-Nets: Neural Networks over Encrypted Data". In: *CoRR* (2014).

[Yao86] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets". In: SFCS '86 (1986).

[YLX13] B. Yao, F. Li, and X. Xiao. "Secure nearest neighbor revisited". In: (Apr. 2013), pp. 733–744.

[ZCS20] Martin Zuber, Sergiu Carpov, and Renaud Sirdey. "Towards Real-Time Hidden Speaker Recognition by Means of Fully Homomorphic Encryption". In: *Information and Communications Security*. Ed. by Weizhi Meng, Dieter Gollmann, Christian D. Jensen, and Jianying Zhou. Cham: Springer International Publishing, 2020, pp. 403–421.

[ZM06] J. Zhan and S. Matwin. "A Crypto-Based Approach to Privacy-Preserving Collaborative Data Mining". In: (Dec. 2006), pp. 546–550.

[ZS21] Martin Zuber and Renaud Sirdey. "Efficient homomorphic evaluation of k-NN classifiers". In: *Proceedings on Privacy Enhancing Technologies* (2021).

[ZYC16] Qingchen Zhang, Laurence T. Yang, and Zhikui Chen. "Privacy Preserving Deep Computation Model on Cloud for Big Data Feature Learning". In: *IEEE Trans. Computers* 65 (2016).

[ZZX09] Feng Zhang, Gansen Zhao, and Tingyan Xing. "Privacy-Preserving Distributed k-Nearest Neighbor Mining on Horizontally Partitioned Multi-Party Data". In: (2009). Ed. by Ronghuai Huang, Qiang Yang, Jian Pei, João Gama, Xiaofeng Meng, and Xue Li, pp. 755–762.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

**Titre:** Confidentialité des données de l'apprentissage machine par chiffrement homomorphe

**Mots clés:** Cryptographie, FHE, Apprentissage Machine

**Résumé:** L'objectif de mes travaux tout au long de cette thèse a été de permettre à des algorithmes complexes d'apprentissage machine de pouvoir être appliqués (lors de leur phase d'inférence) sur des données dont la confidentialité est préservée. Un contexte d'application est l'envoi de données sur un serveur distant sur lequel un algorithme est évalué. Selon les cas, pour des raisons éthiques, légales ou commerciales, la confidentialité des données qui sont envoyées doit pouvoir être respectée. Il est possible pour cela de désigner des autorités en lesquels tous les acteurs du protocole peuvent avoir confiance. Pourquoi accorder à des entités un tel niveau de confiance dans des cas où la confidentialité des données d'un utilisateur est essentielle ? La cryptographie offre en effet des alternatives, dont le chiffrement totalement homomorphe.

Le chiffrement homomorphe permet, en théorie, l'évaluation de n'importe quelle fonction dans le domaine chiffré. Son utilisation peut donc être imaginée dans le cas ou un utilisateur envoie des données chiffrées sur un serveur distant qui détient un algorithme puissant d'apprentissage machine. La phase d'inférence de cet algorithme est alors effectuée sur donnes chiffrées et le résultat est renvoyé à l'utilisateur pour déchiffrement. La cryptographie propose d'autre méthodes de calcul sur données chiffrées qui sont présentées succintement dans le manuscrit. Pour faire court, la particularité du chiffrement homomorphe est qu'il ne nécessite aucune intéraction entre l'utilisateur et le serveur.

Dans ma thèse, je présente trois principaux algorithmes d'apprentissage machine sécurisés: une évaluation sur données et modèle chiffrés d'un réseau de neurone récursif et discret, le réseau de Hopfield; une reconnaissance de locuteur sur modèle chiffré pour un réseau auto-encodeur, le système VGGVox; une évaluation sur données chiffrées d'un classifieur des k plus proches voisins (ou classifieur k-NN). Notamment, notre classifieur k-NN sécurisé est le premier tel algorithme évalué de manière totalement homomorphe. Nos travaux ouvrent de nombreuses perspectives, notamment dans le domaine de l'apprentissage collaboratif sécurisé.

**Title:** Contributions to data confidentiality in machine learning by means of homomorphic encryption

**Keywords:** Cryptography, FHE, Machine Learning

**Abstract:** We aim to provide a set tools allowing for machine learning algorithms to yield their intended results while ensuring confidentiality properties are achieved for the underlying data. This can be achieved through regulatory measures such as prohibiting the use of a sensitive database in certain cases and restricting its access to certain law enforcement agencies. The fundamental reason for the existence of our work - and every other work like it - is the following: why trust that an outside entity will not misuse personal data when you can have assurances of that fact ? This applies both in the case of a private company that may use/sell your data for profit, legally or illegally. It also applies to use by a government which may or may not have the proper safeguards against abuse, as well as the proper security for data storage and access. In our case, we provide such confidentiality properties though the use of Fully Homomorphic Encryption (FHE). Precisely, most of our work focuses on finding new algorithms for secure outsourced machine learning evaluation using FHE. While other privacy and confidentiality preserving methods are touched upon briefly, we focused our research on homomorphic encryption and strive to explain our choice and its general context. We present three main novel secure machine learning applications: a confidentiality-preserving recursive discrete neural network; a model-confidential embedding-based neural network; a confidentiality-preserving k-NN classifier. Notably, our secure k-NN classifier is the only such algorithm in the literature obtaining a result non-interactively. We evaluate the accuracy and efficiency of these three applications on real-world machine learning problems. We show that our secure schemes compare very favorably to their non-secure counterparts in terms of accuracy, while still running in realistic time. Beyond these schemes themselves, this thesis promotes a specific research direction for secure machine learning. We argue for less (though still some) focus on deep convolutional neural networks and show that looking at somewhat lesser known machine learning algorithms can yield promising results.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
91190 Saint-Aubin, France