Secure Machine Learning by means of Homomorphic Encryption and Verifiable Computing

Abbass Madi

université PARIS-SACLAY

Secure Machine Learning by means of Homomorphic Encryption and Verifiable Computing

Apprentissage machine sécurisé à l'aide de chiffrement homomorphe et de calcul vérifiable

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, Institut LIST, 91191, Gif-sur-Yvette, France
Graduate School : Informatique et sciences du numérique. Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche **LIST** (Université Paris-Saclay, CEA) sous la direction de **Renaud Sirdey**, directeur de recherche au CEA List et le co-encadrement de **Oana Stan**, chargée de recherche au CEA List

Thèse présentée et soutenue, le 10 mars 2022, par

Abbass Madi

Composition du Jury

Caroline FONTAINE Directrice de Recherche CNRS, ENS Paris-Saclay	Présidente
Philippe GABORIT Professeur des Universités, Université de Limoges	Examinateur
Dario Fiore Associate Research Professor, IMDEA	Rapporteur & Examinateur
Melek Önen Maître de Conférence, HDR, EURECOM	Rapporteur & Examinatrice
Renaud Sirdey Directeur de Recherche CEA, CEA Paris-Saclay	Directeur de thèse

THESE DE DOCTORAT

NNT : 2022UPASG019

université PARIS-SACLAY

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

Titre : Apprentissage machine sécurisé à l'aide de chiffrement homomorphe et de calcul vérifiable.

Mots clés : Cryptographie, chiffrement homomorphe, calcul vérifiable et Apprentissage machine.

Résumé : L'apprentissage automatique (ou le Machine Learning) est un domaine scientifique très en vogue en raison de sa capacité à résoudre les problèmes automatiquement et de son large spectre d'applications (par exemple, le domaine de la finance, le domaine médical, etc.). Les techniques de Machine Learning (ML) ont attiré mon attention du point de vue cryptographique dans le sens où les travaux de ma thèse ont eu comme objectif une utilisation sécurisée des méthodes de ML.

Cette thèse traite l'utilisation sécurisée des techniques de ML sous deux volets : la confidentialité des données d'apprentissage ou des données pour l'inférence et l'intégrité de l'évaluation à distance des différentes méthodes de ML. La plupart des autres travaux traitent que la confidentialité des données et que pour la phase d'inférence.

Dans ma thèse, j'ai proposé trois architectures pour assurer une évaluation à distance sécurisée pour les configurations suivantes de ML: la classification à distance grâce à un réseau de neurones (NN), l'apprentissage fédéré (FL) et l'apprentissage par transfert (TL). Notamment, les architectures pour l'apprentissage fédéré et l'apprentissage par transfert sont les premiers approches qui traitent à la fois la confidentialité de données et l'intégrité du calcul. Ces architectures ont été construites en utilisant ou en modifiant un schéma de calcul vérifiable pré-existant pour des données chiffrées en homomorphe. Nos travaux ouvrent des nombreuses perspectives, qui ne concernent pas forcément que les architectures de ML, mais aussi les outils utilisés pour assurer les propriétés de sécurité. Par exemple, une perspective importante est de rajouter de la confidentialité différentielle (DP) à notre architecture d'apprentissage fédéré.

Title : Secure Machine Learning by means of Homomorphic Encryption and Verifiable Computing.

Keywords : Cryptography, Homomorphic Encryption, Verifiable Computing, and Machine Learning.

Abstract : Machine Learning (ML) represents a new trend in science because of its power to solve problems automatically and its wide spectrum of applications (e.g., business, healthcare domain, etc.). This attractive technology caught our attention from a cryptography point of view in the sense that we worked during this Ph.D. to ensure secure usage of ML setups.

Our Ph.D. work proposes a secure remote evaluation over different ML setups (for inference and for training). This thesis addresses two cornerstones: confidentiality of training or inference data and remote evaluation integrity in different ML setups (federated learning or transfer learning-based inference). In contrast, most other works focus only on data confidentiality.

In our thesis, we proposed three architectures/frameworks to ensure a secure remote evaluation for the following ML setups: Neural Networks (NN), Federated Learning (FL), and Transfer Learning (TL). Particularly, our FL and TL architectures are the first that treat both the confidentiality and integrity security properties. We built these architectures using or modifying pre-existing VC schemes over homomorphic encrypted data: mainly we use VC protocols for BFV and Paillier schemes. An essential characteristic for our architectures is their generality, in the sense, if there are improvements in VC protocols and HE schemes, our frameworks can easily take into account these new approaches. This work opens up many perspectives, not only in privacy-preserving ML architectures, but also for the tools used to ensure the security properties. For example, one important perspective is to add differential privacy (DP) to our FL architecture.

Contents

Synthèse en français

L'apprentissage automatique (ou le Machine Learning) est un domaine scientifique très en vogue en raison de sa capacité à résoudre les problèmes automatiquement et grâce à son large spectre d'applications (par exemple, le domaine de la finance, le domaine médical, etc.). Les techniques de Machine Learning (ML) ont attiré mon attention du point de vue cryptographique dans le sens où les travaux de ma thèse ont eu comme objectif une utilisation sécurisée des méthodes de ML.

Le besoin d'une grande quantité des données pour entraîner un modèle de ML pose des différents questions (par exemple juridiques, éthiques, commerciales) et, en plus, l'apprentissage collaboratif et les services à distance ouvrent la voie à de nouveaux cybermenaces. Cela motive les approches de ma thèse qui contribuent à ce domaine en utilisant ou en construisant des outils cryptographiques pour profiter de la puissance des modèles de ML tout en garantissant certains objectifs de sécurité. A ce titre, notre travail propose une évaluation sécurisée, à distance de différentes méthodes pour les algorithmes de ML (à la fois pour la phase d'inférence et pour l'apprentissage). Cette thèse traite l'utilisation sécurisée des techniques de ML sous deux volets : la confidentialité des données d'apprentissage ou des données pour l'inférence et l'intégrité de l'évaluation à distance dans les différentes méthodes de ML. La plupart des autres travaux traitent que la confidentialité des données et que pour la phase d'inférence.

Une alternative pour assurer la confidentialité de données est le chiffrement homomorphe (Homomorphic Encryption - HE) qui permet théoriquement d'évaluer n'importe quelle fonction sur des données chiffrées. En complément, le calcul vérifiable (en anglais, Verifiable Computation – VC) permet de prouver à un vérificateur l'exactitude d'évaluation de la fonction déléguée à un proveur. Le VC permet donc d'assurer l'intégrité pour l'évaluation d'une fonction déléguée à un serveur non fiable. En outre, le chiffrement homomorphe et le calcul vérifiable utilisés dans ma thèse sont non-interactives, c.à.d. aucune interaction entre l'utilisateur (le vérificateur) et le serveur (le proveur) n'est demandée.

Dans ma thèse, j'ai proposé trois architectures pour assurer une évaluation à distance sécurisée pour les configurations suivantes de ML : la classification à distance grâce à un réseau de neurones (NN), l'apprentissage fédéré (FL), et l'apprentissage par transfert (TL). Notamment, les architectures pour l'apprentissage fédéré et l'apprentissage par transfert sont les premiers approches qui traitent à la fois la confidentialité de données et l'intégrité du calcul. Ces architectures ont été construites en utilisant ou en modifiant un schéma de calcul vérifiable pré-existant pour des données chiffrées en homomorphe. Plus précisément, j'ai utilisé les protocoles de VC pour les schémas homomorphes BFV et Paillier. Une caractéristique essentielle de ces architectures est leur généralité, c.à.d.

s'il y a des améliorations futures pour le calcul vérifiable ou le chiffrement homomorphe, nos architectures restent valides et pourront très facilement intégrer des primitives cryptographiques nouvelles.

Nos travaux ouvrent des nombreuses perspectives, qui ne concernent pas forcément que les architectures de ML, mais aussi les outils utilisés pour assurer les propriétés de sécurité. Par exemple, une perspective importante est d'ajouter de la confidentialité différentielle (DP) à notre architecture d'apprentissage fédéré et de concevoir des protocoles de calcul vérifiable améliorés qui ne sont pas limités par le degré de la fonction déléguée. Aussi, l'un de nos objectifs était que nos architectures soient suffisamment générales pour fonctionner dans différents domaines et applications et donc une des perspectives vise leur application dans de nouveaux secteurs d'activité.

Acknowledgments

Firstly and foremost, I would like to express my sincere thanks, appreciation, and deepest gratitude to my supervisors, Professor Renaud Sirdey and Oana stan, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level, and especially to stay with me against different difficulties. I mainly have you to thank for that, your trust and your advice. Looking forward to working together with you some more.

I would like to extend my sincere thanks to Melek Önen and Dario Fiore for reviewing this thesis. Your encouraging words and thoughtful, detailed feedback have been very important to me. I realize the amount of your work. For this and their kind words in their reports, and 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 Philippe Gaborit, a member 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.

I am fortunate to have been a part of the CEA teams, I would like to acknowledge my colleagues in the LCYL lab for their wonderful collaboration. To the guys at the DM2I department -Aurélien Mayoue, Arnaud Grivet Sébert and Cédric Gouy-Pallier - thanks for the opportunity to work with you. A special thanks to my friends Mohamad Amhaz, Ali Mokh, Moussa kafal, Alaa Ali Hasan, Fatima Moustafa, Abbass Taki, Yousef Boukhair, Zahraa Alayan.

In addition, I would like to express my gratitude and love to the most valuable people of my life, my family. I would like to thank my parents who provided me with love and support in every possible way. This day would not have been possible without them and no words can express my thank and gratitude to you. For my parents-in-law, for your support, for your wise counsel, and for your sympathetic ear, thank you very much! For the support of my wife's family -Mohamad, Alaa, Ali - thank you. A special thanks to all members of my family-Ali, Hala, Imad, Jihad, Maged, nour- especially my sister Fatima and my brother Khalil. This is in great part thanks to you. For Khalil, I have looked up to your spirit of research and your determination all my life, I have always benefited from our discussions and your support, motivation for me.

A big thanks to the love of my life, my wife, Batoul Awada. she has been extremely supportive of me throughout this entire process and has made countless sacrifices to help me get to this point. You provided happy distractions to me to rest my mind outside of my research and thank you for your support and patience. Now that this is done, we can

expect many more challenges ahead and I look forward to our common struggles.

Of course, I can't name everyone. Thank you everyone I may have missed in these acknowledgments.

Finally, I thank my God, for letting me through all the difficulties. You are the one who let me finish my degree. I will keep on trusting You for my future. Thank you, Lord.

Chapter 1

Introduction

1.1 General Scene

Facebook–Cambridge Analytica data scandal . On March 25, 2018, Marck Zuckerberg, the CEO of Facebook, published a personal letter in various newspapers apologizing on behalf of Facebook about the privacy violation of about 87 million Facebook users. Cambridge Analytica has used Facebook users' data from different applications via "Facebook's Open Graph platform". These data were used to provide analytical assistance to the 2016 presidential campaigns of Ted Cruz and Donald Trump, and it was accused of intervening with the Brexit referendum. The misuse of these data was disclosed in 2018 by a former Cambridge Analytica employee Christopher Wylie in interviews with the New York Times and the Guardian journal. Eventually, Facebook apologized for its role in data harvesting and underwent Zuckerberg to testify in front of the United States Congress.

According to the New York Times[1], the data collected by Facebook is detailed information that can permit Cambridge Analytica to create psychographic profiles of the subjects of the data, that included also the personal location, which is considered to be a very sensitive information.

Today, information and data privacy is the preoccupation of institutions, media, and the awareness media. Different ethical and legal questions were raised about the data use. For example, here are some questions concerning the access and storage of the user's data: does one own their data, where this data is stored, can the provider service (such as Facebook, Amazon) sell this data or give access without an immediate effect to friendly and/or family zone?

[1] Journal available at https://www.nytimes.com/2018/03/17/us/politics/cambridge-analytica-trump-campaign.html.

An essential property in the remote outsourcing evaluation is the user's autonomy. The users' autonomy in the outsourcing evaluation over users' data means obtaining the users' consent after receiving information to understand: the purpose, the risks, and the methodology of the research being conducted over their data. Giving access to the users' data may have repercussions for all who share an essential part of the users' private life. Another threat in the outsourcing evaluation is the case of hacked data, where the users' cannot access or control their data.

Data mining is a growing research and technology field since machine learning techniques and AI models are being progressively deployed in different activity sectors (e.g., health, finance, autonomous vehicles, energy, etc.). In this context, the ethical, legal-related questions become crucial (especially those that concern privacy) and more urgent to be addressed. This is particularly true in countries where governments don't care about data protection and personal privacy.

An example of an interesting Machine Learning (ML) application that needs Big Data is the medical domain (like cancer prediction), which requires large volumes of training data to obtain reliable predictions. In this case, one would desire to contribute with their personal information in the training phase of an ML system because of its lofty goals. Still, one remains apprehensive about her/his data security, especially when her/his government turns a blind eye to these securities of data.

After the Facebook-Cambridge scandal, in July 2019, Federal Trade Commission announced that Facebook was fined $ 5 billion due to privacy violations. In the UK, in October 2019, Facebook agreed to pay a $500,000£$ fine to the UK Information Commissioner's Office for exposing the data of its users to a "serious risk of harm" according to BBC NEWS[2].

In the European Union, the legal and ethical issues related to the data place and its storage are being solved with more consideration than in other countries. On 25 May 2018, the EU approved the General Data Protection Regulation (GDPR), which governs how the personal data of an individual in the EU could be processed and transferred. The GDPR has a tiered penalty structure that will take a significant bite out of offenders' funds and the EU GDPR rules apply to both data controllers and processors. A non-compliance to the EU GDPR rules results in fines up to 4% of the global revenue. Yet, we cannot rely on Non-Governmental Organization (NGOs) and governments to ensure data privacy, integrity, and confidentiality since each has its goals and instruments. Moreover, although their good intentions are assumed, there are doubts about the ability of countries to impose control or ensure data security in the digital world. There are companies whose revenues are more significant than the global revenue for one country. At the time of writing this thesis, the market value of Apple (Apple Inc) become $ 2.703 trillion in December 2021[3]. It exceeds the total foreign exchange reserves (Forex reserves) of 4 countries: Germany, France, the UK, and Italy, which is about $ 0.861 trillion[4] of resources on August 2021.

These ethical and legal questions motivate a growing body of research to solve them and to develop new tools for improving data security. Then one can choose the suitable tools

[2]https://www.bbc.com/news/technology-50234141
[3]https://investor.apple.com
[4]https://tradingeconomics.com/country-list/foreign-exchange-reserves

to solve their specific issues. It is also necessary to research this topic to improve the existing tools or discover new tools to solve pre-existing or newly emerging issues.

What do we search for? This work attempts to propose frameworks that permit one to use their data in ML models with a more significant security level. Especially, we preserve two broad security properties: integrity and confidentiality, and, more precisely, we ensure data privacy and evaluation integrity. For example, it permits one to contribute with their data in medical machine learning training or to analyze their medical data without worrying about the data security, when the training or analysis are outsourced.

The following questions could be asked: Why does one trust a third party to access their data, calculate any function over their data without any guarantees for the correctness of outsourced computation and the confidentiality of their data. Instead, this third party can sell the users' data and analyze it to achieve a specific goal. But, one can achieve their own goal and ensure their aimed security without using this third party. Imagine that this is applicable with different companies that someone shares their data with. At the same time, the government cannot ensure the users' data security against misuse, dissemination, and storage in a service provider (SP). Moreover, the users don't trust a third party, but they want to achieve a special benefit from their use of SP resources as a large scale computation provider, and decrease the cost of storing data.

In this thesis, we strive to describe the types of threats for the outsourced computation in the case of malicious servers and we describe the tools to ensure security for the users data and calculation against the threats, and how to adapt these tools or improve them to solve larger threat models. To address wider these threats there are many ways and, each of them has to resolve a specific threat question like: what are we protecting? Against who? What is the type of calculations delegated? What is the power of the provider in the calculation? These questions and other ones will increase in the years to come especially with the appeareance of a new type of service: the computation as a commodity, which is the natural consequence of the increasing popularity of the cloud computing paradigm.

1.2 Cryptography

Cryptography, also known as science of the information security, is the study of methods to hide the information against a third party. The legend says that the first known cipher was used in an ancient war a long time ago, where a squad leader shaves the element hair, writes sensitive information on his head, and waits for the growth of its hair. Then, when it was done, he sent the member inside the groups of enemies with this information to pass to another sector of this squad to convey the information. Regardless of the reality of this story, it is enough to give us an impression of the importance of cryptography in history. By our uses of the internet and other connected devices in our daily lives, we are aware of the importance of this branch of research to improve our daily life.

More formally, cryptography defines various secure methods to share information and keep it secret when communicated between two or more parties over insecure channels in such a way where only those for whom the information is intended can read and process it. Cryptography is used to ensure secure and safe communication between these parts in the presence of third parties called adversaries.

In the beginning, suppose that Alice (A)- as per tradition- wants to send a clear (or plaintext) message m to Bob (B) over an insecure channel. She is going to use a cryptosystem to transmit m to Bob. Namely, Alice runs the encryption process, i.e., transforming message plaintexts into ciphertexts c under the use of a secret key, known only to Alice. This is realized such that no one should find the original plaintext without knowledge of the secret key. Then Alice shares sk with Bob on a secure channel (in person, for instance) and sends the ciphertext in front of everyone. Now, Bob can use the sk and the ciphertext received to retrieve the plaintext that Alice intended to share with him using an algorithm called decryption. These methods are part of the large body of symmetric encryption cryptographic methods.

The internet deployment and its different uses opened the Pandora's box of digital security ills, which increases the need to develop more versatile cryptographic systems. Many cryptosystems appeared and this branch of sciences occupied its unique place in the interests of governments and militaries.

Therefore, several cryptosystems have been used over the centuries, starting from the Caesar cryptosystem to different cryptosystems today. For a long time, the cryptosystem held symmetric keys; that is, the encryption and decryption algorithm use the same key until the emergence of the public-key encryption schemes [1] (also known as asymmetric encryption), which began a revolution in cryptography sciences. Later, the hybrid cryptosystem surfaced that combined the use of asymmetric encryption to exchange the key of symmetric encryption and use this key in a symmetric cryptosystem. Today the cryptography branches contain different methods to ensure security for different applications like digital signature, key management, and message authentication.

1.3 Thesis Technical Scene

The scope of our work is to ensure a secure evaluation for different outsourced machine learning setups. We focus on the two core bases, information security methods, namely: the integrity and the confidentiality, using two cryptographic tools: *Homomorphic Encryption (HE)*, and *Verifiable Computing (VC)*. In the following, we describe the terms related to what is defined above. It is worth noting that these terms will be well presented in this thesis.

Homomorphic Encryption is a corpus of cryptographic techniques that ensure data-confidentiality. It evolved significantly in the last twenty years and it is one of the principal tools used in this thesis. Specifically, it allows the computation directly over encrypted data without decrypting them. Therefore, HE ensures data confidentiality while data are exchanged and while a non-secure-enough platform processes them. The scheme that accepts the calculation of any arbitrary function is known as *Fully homomorphic Encryption* (FHE). Sadly, to this day, there are no FHE schemes that calculate any arbitrary function in an acceptable time. In this research, we will define homomorphic encryption, particularly the schemes employed in our work - BFV, BGV, Paillier- in much more prominent details in chapter 3.

Verifiable computing comprises methods that support the delegation of the computation for a function on outsourced data to third parties, such that the data owner and other users can verify that the third party has computed the outcome result correctly.

Consequently, it is used to ensure data integrity, with malicious servers that seek to breach the data users' security.

Machine Learning (ML). Our work does not address the Machine Learning core but the Machine Learning applications. We intend to ensure a secure and efficient corpus of methods/frameworks to evaluate machine learning methods securely. Specifically, we focus on the two principal phases of a ML algorithm: either to provide a secure training phase or to securely evaluate the inference phase. Generally, there exist different building structures for the ML algorithm one has to protect against an intruder's eyes, such as 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), and the data used for the training and prediction, etc. Each of these elements poses its proper challenges. In this work, we discuss providing security of the data and training model. In terms of techniques, we present two secure classification algorithms of machine learning. Although there exist other ML techniques (clustering, regression, etc.) the classification is one of the most common and most used techniques.

Security: At first look, one may think it is a simple concept. However, that simplicity quickly fades, particularly in the cryptography domain, where the security concept is linked to several aspects mentioned earlier, like what do we want to secure? Against what? What is the power of the adversary?.. This *security* can be ensured using different methods such as material solutions or algorithmic solutions. To be precise, in this work, essentially, in the outsourced evaluation of ML algorithm on a cloud (or external server), we study three different setups (inference, federated learning and inference using transfer learning), where we will take into account the different threats of security. Notably, in this work, we detail the data confidentiality concept of outsourced computation and the integrity of treatment (which includes the all/part of training or prediction algorithm) with different uses cases.

1.4 Our Contribution

This thesis addressed the two cornerstones: confidentiality and integrity in the use of different setups and/or phases of machine learning methods. To achieve this goal, we build three architectures/frameworks to ensure a secure evaluation for the following ML methods: Neural Networks, Federated Learning, and Transfer Learning. We tweak and use existing FHE schemes and VC schemes to our convenience. Therefore, we can summarize our contributions as follows:

- **A secure Neural Network Evaluation** The first contribution is described in chapter 5. It was published as [2] and presented at the CLOUD S&P 2020 workshop. This work proposes a practical framework for privacy-preserving predictions with Homomorphic Encryption (HE) and Verifiable Computing (VC). We designed a partially encrypted Neural Network in which the first layer consists of a quadratic function and its homomorphic evaluation is checked for integrity using a VC scheme which is a slight adaption of the one of Fiore et al. [3]. Inspired by the neural network model proposed by Ryffel et al. [4] which combines adversarial training and functional encryption for partially encrypted machine learning, our solution can be deployed in different applications contexts and provides additional security guarantees. We validate our work on the MNIST handwritten recognition dataset

for which we achieve high accuracy (97.54%) and decent latency for a practical deployment (on average 3.8 seconds for both homomorphic evaluation and integrity proof preparation and 0.021 seconds for the verification).

- **A Secure Federated Learning framework** This is the first Federated Learning (FL) framework, which is secure against both confidentiality and integrity threats from the aggregation server, in the case where the resulting model is not disclosed to the latter. We do so by combining Homomorphic Encryption and Verifiable Computing techniques to perform a Federated Averaging operator directly in the encrypted domain (using HE) and produce formal proofs that the operator was correctly applied (using VC). Due to the aggregation function's simplicity, we can ground our approach in additive HE techniques, which are highly mature in terms of security and decently efficient. We also introduce several optimizations, which allow reaching practical execution performances on the larger deep learning models end of the spectrum. Also, we provide extensive experimental results on the FEMNIST dataset, demonstrating that the approach preserves the quality of the resulting models at the cost of practically significant computing and communication overheads. At least in the cross-silo setting, higher-end machines can be involved on both the client and server sides.

- **A Secure Transfer Learning Architecture** This is the last contribution presented in detail in chapter 7, where we investigate the possibility of realizing complex machine learning tasks over encrypted inputs with guaranteed integrity. Our approach combines Fully Homomorphic Encryption (FHE) and Verifiable Computing (VC) to achieve these properties. To work around the practical difficulties when using these techniques - high computational cost for FHE and limited expressivity for VC- we leverage on transfer learning as a mean to (legitimately) decrease the footprint of encrypted domain calculations without jeopardizing the target security properties. In that sense, our approach demonstrates that scaling confidential and verifiable encrypted domain calculations to complex machine learning functions does not necessarily require scaling these techniques to the evaluation of large models. We furthermore demonstrate the practicality of our approach on an image classification task.

1.5 Manuscript Overview

Let us now describe, chapter by chapter, the content of this manuscript. Generally, this manuscript consists of two parts. One introduces the background and the technical overview surrounding our work, and the other specified our scientific contributions to that background.

Part I presents the context and the state of the art of the landscape in which our work was realized. We present in chapter 2 the context and motivation of our work for this thesis focusing on machine learning and security guarantees to maintain confidentiality and integrity. Specifically, we specify the position of our work in this field, in addition, to helping the reader follow our choices and the limitations we faced. Chapter 3 is dedicated to the introduction of the technical notions that we use throughout the thesis. It highlights the FHE schemes that are used in our work, with the global goal to present a broad overview of the primary scientific background of these schemes. In chapter 4, we present

the technical background for verifiable computing. It also covers more in details the VC schemes based on fully homomorphic encryption.

Part II consists of three chapters, each of which introduces one of our contributions. We note that each one of these chapters is an article that has been published (chapter 5 and chapter 6) and accepted paper (chapter 7). Chapter 5 intended goal is to present our secure training of neural networks using homomorphic encryption and verifiable computing. Chapter 6 introduces our framework to evaluate secure federated learning using verifiable computing on Pallier encryption scheme. Chapter 7 presents our architecture for a secure evaluation of the transfer learning technique.

1.6 Publication and Talks

Published Papers

- A. Madi, R. Sirdey, and O. Stan. "Computing Neural Networks with Homomorphic Encryption and Verifiable Computing". International Conference on Applied Cryptography and Network Security. Springer, Cham, 2020.
- A. Madi, O. Stan, A. Mayoue, A. Grivet-Sébert, C. Gouy-Pailler and R. Sirdey."A Secure Federated Learning framework using Homomorphic Encryption and Verifiable Computing", 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS), 2021, pp. 1-8.
- A. Madi, O. Stan, R. Sirdey and C. Gouy-Pailler. "SecTL: Secure and Verifiable Transfer Learning-based inference", 2022 International Conference on Information Systems Security and Privacy provides (ICISSP).

Talks

- A secure approach for Neural Networks using Homomorphic Encryption and Verifiable Computation, Journées Codage & Cryptographie, May 2020.
- Computing Neural Networks with Homomorphic Encryption and Verifiable Computing. International Conference on Applied Cryptography and Network Security, October 2020.
- A Secure Federated Learning framework using Homomorphic Encryption and Verifiable Computing, 2021 Reconciling Data Analytics, Automation, Privacy, and Security (RDAAPS), May 2021.

Part I

Context and state of the art

Chapter 2

Context and Motivation

2.1 Machine Learning

Machine Learning is a branch of AI (Artificial Intelligence) and Computer Science which focuses on the study of algorithms that give a computer the power to solve a problem automatically without any explicit information about how to solve it. Namely, it automatically learns programs from data. It was born in 1950s and recently evolved to reach a level of public attention and industry investment never seen before in the history of Artificial Intelligence (AI), especially due to the use of Deep Convolutional Neural Networks.

Supervised Learning is by far the most common approach in machine learning, used in applications like image classification, speech recognition, optical character recognition, and language translation. It mostly consists of classification and regression category, where a regression algorithm (like linear, logistic) is used to predict continuous values and a clafssification algorithm (like: linear classifiers, support vector machines, decision trees and random forest) is used to predict/classify discrete values.

A supervised machine learning lifecycle can be divided into two main parts: a training phase and an inference phase. The difference between them is described as follows:

- *Training* phase: This is the first phase, in which a model is created and/or trained over data. The training starts by dividing the data into a training set and a testing set, where the trained model uses the training data, and the test set is used to evaluate the model once the training is complete. The success rate obtained after training by an evaluation over the testing set determines the accuracy of the machine learning algorithm.
- *Inference* phase: After the learning is complete, the model is put into action on live data to classify/predict an actionable output for a real-world application. This process is also referred to as "operationalizing a ML model" [5]. Figure 2.1 shows the training and the inference phase of a standard machine learning system.

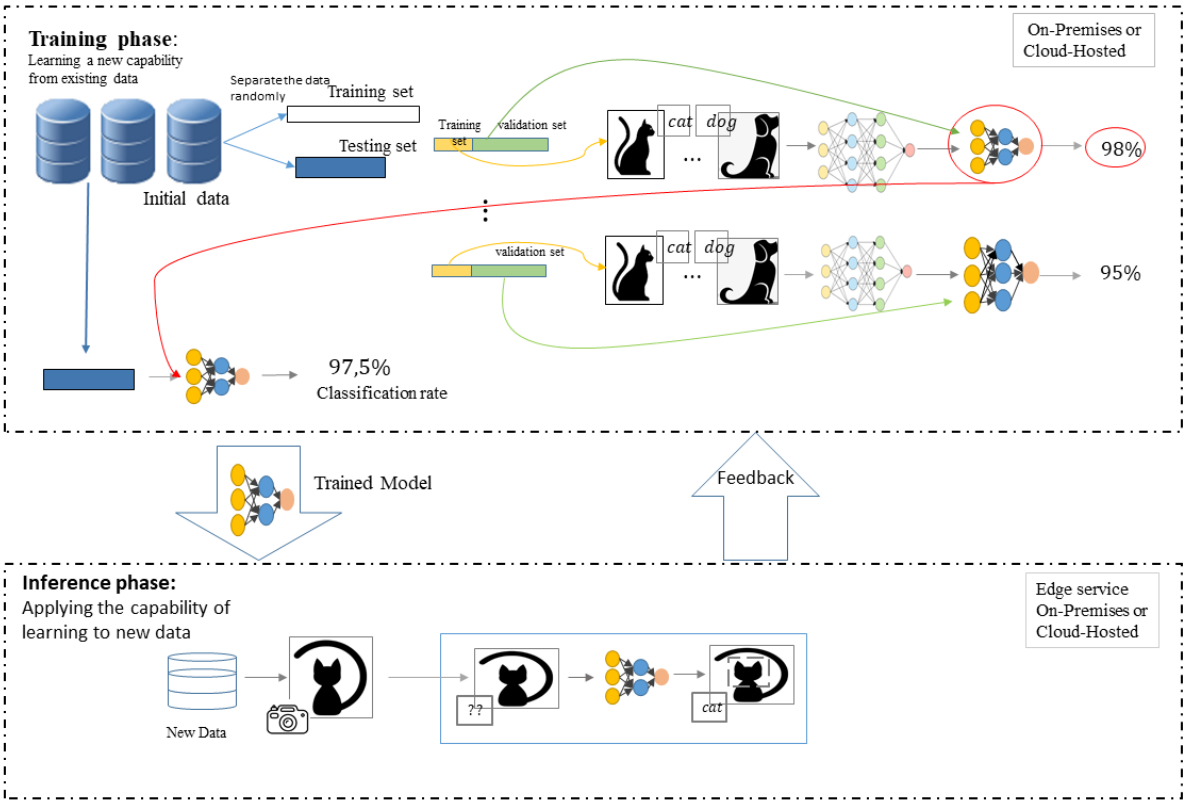

Figure 2.1: This figure represents the different classical steps of training and inference phases of an ML system. First, in the training phase: the data scientist separates the dataset into training (white) and testing set (blue) where the testing set will never be used during the training process. Different parameters (for example weights of neural network) for the model are created using several randomized sets of training data (yellow) with an unchanging training algorithm. We compare these parameters by applying them over the validation set (green), the best parameters are selected and the corresponding model is tested for classification accuracy over the testing set. The inference phase evaluates the selected model over new data. In some cases, if the model permits it, the result from the inference phase is fed back to the training phase for a continual learning process.

In machine learning, we use the feedback process if the machine learning algorithm permits it, where the machine learning model's inference outputs are fed back to the model for a

constant learning process to improve the achievement in learning.

The data is an essential component of any machine learning model and it is the base of any machine learning model to the extent that we can say that the data is more important than the algorithm itself. Over time, the required data to train a machine learning model increased and this is even truer with the inception of convolutional neural networks and deep learning algorithms. Companies and governments work to access big data to perform deep learning algorithms and data is rather a very valuable asset for that.

The goal of this work is not machine learning per se, but to respond to the following question "How can we provide a secure machine learning application?". For this goal, our work is to protect the data and/or the ML model for any user and/or any platform using different techniques of machine learning which are "FHE-VC-friendly[1]". The capabilities of FHE and VC are limited and their application can produce a high modification (size, type..) in the data format and in the machine learning models, a consequence being that we cannot apply any machine learning algorithm we would like.

2.2 Security Threats

In this thesis, we are interested in machine learning where some or all computations are outsourced, and we want to achieve a secure remote machine learning function. Generally, we have three entities, the user/querier, the owner of the data, the server performing the evaluation of the training phase and/or the inference phase, and finally, an operator/client having access to the results.

In terms of ML phases security, the best case is the evaluation of all ML phases securely. However, depending on the use case, we can ensure "secure" training without "secure" inference and vice versa.

The security of remote machine learning algorithms implies multiple aspects, covered by the CIA triad model: Confidentiality, Integrity and Availability, the three main pillars of cybersecurity.

2.2.1 Confidentiality Threats

2.2.1.1 What is confidentiality ?

We can consider that confidentiality and privacy definitions are at the heart of the ethical and information security research. Often in the literature on information security the two notions are indistinguishable and they are defined as ways of "keeping information secret from all but those who are authorized to see it" according to [6]. In the context of outsourcing computation, we can distinguish between confidentiality and privacy, where confidentiality means the information are kept hidden, while on the other hand the privacy means the inability to detect a correlation between an entity and its personal data.

As we see above, machine learning solutions are widely employed in different domains to improve real life. To make this solution perform better (high training/testing accuracy),

[1]FHE-VC-friendly means that we can evaluate this models over homomorphic encrypted data with VC existing scheme and with practical performances.

you need a large dataset, which poses a problem with these data, in particular with the confidentiality of these data.

Data confidentiality has become a primary concern for citizens and governments. Often, the data owner does not allow anyone to use his/her private databases because the risks of data disclosure are too great (like the Facebook–Cambridge Analytica data scandal). Also, the collection of data can be prevented by society or governments in the name of confidentiality and privacy.

Therefore, this work focuses on preserving the confidentiality and privacy of the data exploited by an outsourced computation machine learning algorithm that is important for different kinds of reasons: legal, financial, and ethical issues. Outsourcing the training and/or testing of machine learning models drives to understand the different confidentiality and privacy properties that one desires to achieve. To illustrate these properties and the related threats, we present an example of outsourcing medical machine learning computing.

Consider the case of a health-related database, which reveals highly confidential personal information. This database is collected from personal records (e.g., multi-dimensional vectors) and stored on a secure server. Each record is composed of several attributes (e.g., the element of the vectors) and contains personal or medical information (medical records case). In this context, a client -an outside entity- wishes to train, analyze/diagnose their data with a remote machine learning model that requires an input: the query (e.g., a multi-dimensional vector made up of attributes) without worrying about their data security. More formally, the client wants to achieve the secure use of her/his data during the outsourcing storage, computing, or in a remote learning process. There are some confidentiality/privacy properties presented in [7] that allow ensuring the client's requirements that are presented below and enumerated from 1 to 7.

1. Record Confidentiality: No one excepts the authorized readers can read the clear form of the records on the server. Reasons for this can be multiple, from the high cost of acquiring a good database, to the legal or ethical constraints: there is a common-law duty to keep personal data (including medical records) secure. It may not be used or sold by anyone without the confirmation of the principal owner even if that data is not linked to the user from which it was obtained.

2. User Privacy: Just the server can link a given user to its personal data in clear form. It is weaker than record confidentiality as one could achieve user privacy but not record confidentiality. The reverse is not true.

3. Attribute Privacy: Whereby no one can link a single user to the value of an attribute. It is a harder constraint than user privacy.

4. Query Confidentiality: the nature of the query is known only by the client. It can be important in the case where the computation outsourced is a comparison of the client's personal data (the query) with the server's data. Similar to record confidentiality, but with respect to the client.

5. Client Privacy: Whereby no one can have access to both the contents of the query and the identity of the client. It is similar to user privacy, but with respect to the client.

6. Computation confidentiality: Only the server knows the nature of the computation

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.	∼	∼	∼	✓	✓	∼	✓

Figure 2.2: This figure presents different basic protocols with or without FHE and their impact on the privacy and/or confidentiality of the data involved. The ∼ symbol represents a property that is not achieved because of information leakage, the ✗ symbol means the property is trivially not achieved because data is sent in clear form.

that applies to the data. This can be important in the case where the nature of the computation reveals information about a model/algorithm that is expensive to obtain.

7. Result confidentiality: Only the client knows the final result. It can be important in our medical context where the result can be a diagnosis. This diagnosis over the query, which is the client's personal medical data it is a very sensitive information.

There exist basic schemes to solve the problem but each of them achieves different confidentiality requirements (presented in the Figure 2.2 from [7]), but not all of them. In the ideal case, we are interested to ensure all the above properties related to privacy and confidentiality. We will mention (or explain) the tools used to ensure the above confidentiality properties in section 2.4 presenting the countermeasures.

2.2.1.2 Threat analysis

We start by describing two scenarios for medical machine learning models: one for the training phase and the other for the inference phase:

For the training phase, a client like a hospital collects and stores the patient data on a secure server to serve this phase.

For the inference phase, a client like a patient wants to use the evaluation realized by a machine learning model, for medical diagnosis over his/her medical data, which reveals highly confidential information such as the disease history and his/her undergoing treatment.

Therefore, in machine learning, we have three assets, on the data side: the training data

owned by the participants and the user's inference data, on the model side: the machine learning model. The training and inference data must be guaranteed with respect to the threats coming from an adversary that can be the server or another participant or client (regardless of the place of these training and inference phases, i.e. in the same place or different servers). Furthermore, the confidentiality of the model must be guaranteed with respect to the threat coming from the adversary attacking the server on which the model is trained or evaluated. If this is not the case, the server must share his commercially valuable model, or place the entire model in a trusted enclave space, for the client to accept to share his/her information for this machine learning model. This sharing is non-suitable for large models because they lose their competitive advantages, and then the business value.

The focus of this thesis is to ensure confidentiality of training or inference phases of a machine learning algorithm, and this is presented in detail in the following contribution chapters. We note that we treat only the threats coming from the server and not the threats coming from another participant or client of the ML model.

2.2.2 Integrity Threat

In the literature, the integrity is defined in several ways. Moreover, this term has specific ethical and/or legal definitions depending on the country. It is generally indicating a service "ensuring information has not been altered by unauthorized or unknown means" according to [6].Namely it guarantees that a message was not modified during its transmission.

Essentially, there are several desirable integrity goals, depending on the use-case and this can be described as:

- The search/quest to ensure full confidence that the data one is receiving is the actual valid data from the sender is called "data integrity".
- The attempts to achieve that the result calculated by a server is correct, which can be simplified by the verification that the remote execution of a function on an untrusted machine is correct or not, is called "execution integrity".

In this thesis, we are looking at model integrity in the case of outsourcing machine learning computation, where a client verifies that the training/inference steps delegated are calculated correctly.

2.2.2.1 Threat analysis

Let's take the case of the medical machine learning use case, where people accept that the hospitals (or any provider healthiness services) manage their health records. This is important particularly in countries where access to health care is complicated. In this case, the participant in the training phase (like hospitals) sends the patients' medical records to evaluate the training over these records. Finally, a client (depending on the used protocol) can send their medical records to the inference phase to achieve its intent.

Then, on the client-side, it is very important to ensure that the computation delegated in the training and inference phases are correct, especially where the result of inference is a medical diagnosis, very sensitive information. Ideally, in some cases where the client evaluates a specific model before sending her/his data into the training and/or inference

phase, the server or other participants in the learning process want to ensure that the specific client evaluates this model correctly.

In this work, we focus to guarantee the integrity of the training and/or inference steps against threats coming from the server.

2.2.3 Availability Threats

In order for information system to be useful, it must be available to authorized users. In the context of remote machine learning/training evaluation, practical remote evaluation required the protection of timely and uninterrupted access to the authorized space of training and/or inference algorithm to the authorized participants.

The most fundamental availability threats are non-malicious in nature, and include hardware malfunctions, network bandwidth issues and scheduled software downtime.

Cryptography alone cannot do anything with availability threats, for this reason, these threats are beyond the scope of this work.

2.3 Adversaries

As we determined above, when we outsource computation -even partially- we need to ensure different security properties (like the integrity of results). Indeed, suppose that all machine learning phases (training and inference) happen on the same server with locally collected training and classification data. In that case, consequently, confidentiality can be ensured using the network security by the server: ensuring there is no information leakage on the server.

An adversary model is an entity: it may be an algorithm, or it may simply be a series of statements. It is a formalization of an attacker. Typically the goal of the adversary is to disrupt or prevent proper operations of a secure system (e.g. by violating the confidentiality, data integrity, or availability of the system). With regards to the capabilities of an adversary, there are a number of approaches in various fields of computer security that fit within this umbrella.

In our work, we outsource computation -or a part of a computation- to an outsourced entity. Then, we need to ensure the security of this outsourcing, but the security properties (like confidentiality and integrity) required will depend on the level of trust for this entity. For example, if we can trust this entity, achieving security can be reduced to simply ensuring these properties over the transfer channels. Otherwise, additional measures will be needed. This level of trust determines the level of adversarial strengths. Therefore, we need to distinguish between the adversary depending on the adversarial strength as in the literature, where two behaviors of adversaries are typically defined as follow:

1. Honest-but-curious (also known as semi-honest or passive adversary): a legitimate participant in a communication protocol follows the protocol properly but he/she tries to learn as much private information as possible. So, with this type of adversary model, we do not need to address integrity threats.

2. Malicious adversaries (also commonly known as active adversaries), where it is necessary to ensure both confidentiality and integrity threats coming from this adversary.

The malicious adversaries may behave arbitrarily (i.e. execute any computation) for stealing, corrupting, and modifying data, without any specifications, and may compute any function over data instead of the required computation.

Now, take the case of the above example, where a participant in the training phase sends his/her medical records to participate in the training or sends a record finally to the inference phase. In this context, the honest-but-curious server computes correctly the delegated training/inference algorithm, with the exception that it keeps a record of all its intermediate computations to learn whatever from this computation. On the other hand, the malicious server can compute any algorithm and return any result, even if it is a sensitive result as a medical diagnosis.

In this work, we are interested in adversarial server cases. We will go beyond the classic assumption that honest-but-curious servers perform calculations related to machine learning phases (training or inference) to solve this malicious server's confidentiality and integrity threats.

2.4 Countermeasures

2.4.1 Confidentiality-Preserving Tools

Ensuring confidentiality can be done in different technical ways. Before presenting the cryptographic methods, we present the "Anonymization"-type techniques which is a set of confidentiality-preserving methods that can be integrated with the cryptographic approaches to achieve a higher level of confidentiality for an information system, especially useful with remote machine learning protocols.

Anonymization:
Anonymization is in a set/corpus of practical methods for preserving user's privacy, consisting in obfuscating the relationship between a user and its data (name, address, identity number, social security number, etc.) However, many attacks have been proposed against this obfuscation.Take the famous case of Netflix Price, where it was announced a one-million-dollar prize for the purpose of improving its movie recommendation system. To do so, Netflix published anonymous movies ratings taken from 500,000 customers between 1998 and 2005 while assuring customers that this would not harm them as it had been thoroughly anonymized. However, in 2006, Narayanan Arvind, and Shmatikov Vitaly succeeded in de-anonymization this dataset using Internet Movie Database (IMDb) as the source of background knowledge. This result was published in their paper "Robust De-anonymization of Large Sparse Datasets" [8], where they successfully identified the Netflix records of known users. Therefore, it is necessary to provide stronger privacy for individuals whose records are used in an outsourcing machine learning computation.

There are generally two models that propose privacy guarantees that have been widely adopted by the security community and are still the basis for most of the following works:

- **k-anonymity**: This notion was introduced by Sweeny and Samarati in 1998 in [9]. This technique has been studied extensively in the database community to ensure user privacy in big data. To achieve k-anonymity, there need to be at least k individuals in the dataset sharing the set of attributes that might become identifying

for each individual, and each record is indistinguishable from at least $k-1$ other records with respect to certain identifying attributes. This technique can be described as a 'hiding in the crowd' guarantee. But this technique has been shown to lack strong privacy guarantees: for example an adversary can discover the values of sensitive attributes when there is little diversity in those attributes, and if it has background knowledge and this is often the case. These weaknesses led to novel and more powerful privacy techniques: like $\ell-$diversity [10], $t-$closeness [11] and $n-$confusion [12].

- **Differential Privacy (DP)**: DP is a set of methods to achieve data privacy, formally defined by Dwork et al. [13], even if the randomized response mechanism (DP mechanism) appeared for the first time in [14]. Generally, the best way to maintain user privacy is the response in a wrong way to a question according to the fundamental law of information reconstructions [15], that states: "overly accurate answers to too many questions will destroy privacy in a spectacular way". More precisely, DP states that adding a conditional noise (less than a specific level to ensure that the noised data remain usable) in the original data produce an overall less accurate results. The work of Dwork [13] states that the result of computation would not have been different if any individual user had not provided their record. From the point of view of use-cases, a given application can propose a constraint over the budget noise, where another one requires to guarantees less security versus more accurate results. There are numerous variants and extensions that were proposed to adapt DP with respect to different scenarios and attacker models. Desfontaines et al. [16] proposed a classification of DP variants and extensions to provide potential users with solid security guarantees.

Cryptography:

In cryptography, there are three general methods to achieve privacy of outsourcing machine learning computation: Multi-Party Computation (MPC), Fully Homomorphic Encryption (FHE), and Functional Encryption (FE). We present them briefly below.

Multi-Party Computation: It started in the late 1970s with the mental poker problem that searches a solution for the question "How can one allow only authorized actors to have access to certain information while not using a trusted arbiter?". The first design saw the light in the 1980s as a theoretical solution to solve Yao's millionaire problem, where two millionaires wish to know which one of them is richer, without knowing any information about each other's wealth. In general, MPC was designed for privacy-preserving applications, that give multiple entities which don't trust each other, the ability to compute a joint function over their data and arrive at the desired result, preventing any other party from gaining information about anything else. All MPC techniques assumed that there is a reliance on an exchange between the participants (two or more) involved as the computation is going on. In the past decade, MPC has been a very active research area in both theoretical and applied cryptography, with the rise of several efficient MPC schemes. These MPC schemes are designed by means of both generic or specialized protocols, allowing any kind of computation or designed specifically for a given set of computations. The MPC for machine learning application assumes that a participant will be online during the exchange of information and induces high-communication costs of the multi-party computation.

Fully Homomorphic Encryption: The problem of constructing a fully homomorphic

encryption scheme was first proposed in 1978 [17], shortly after the invention of RSA [18] scheme. The research around the HE scheme remained slowly developed until 2009, where the turning point came by Craig Gentry, who put forward the first plausible construction of a FHE scheme [19, 20] using the hardness of some lattice problem with the bootstrapping idea. More details on this in the dedicated section, in chapter 3. The goal of constructing a FHE scheme is to perform public, arbitrary, unbounded computation on encrypted data securely. These three properties mean the following:

- Public computation: The public computation means that one can calculate any operation over the encrypted data without any secret information. More precisely, given an encryption of the value x ($Enc(x)$), and a function f, one can compute an encryption of the evaluation of the function f over the value x ($Enc(f(x))$), on their own non interactively using $Enc(x)$, and a public key. It can be considered as the heart of the FHE scheme, therefore, it is necessary that any FHE scheme verifies this property.

- Arbitrary Computation: A scheme supporting arbitrary computations on encrypted data $Enc(x)$, can compute any function f over $Enc(x)$ to obtain $Enc(f(x))$. A cryptosystem that supports arbitrary computations on ciphertexts is said to be fully homomorphic. For instance, most encryption schemes can be considered *partially homomorphic* schemes in that they allow for some types of computation to be run on encrypted data. Currently, the RSA scheme [21] supports an unlimited number of modular multiplications, and the Paillier cryptosystem [22] supports an unlimited number of modular additions.

- Unbounded computation: A specific type of FHE scheme called leveled FHE scheme like [23, 24] can evaluate arbitrary functions but up to a certain (multiplicative) depth L. To decrypt and obtain a correct message, it is necessary that the error associated with the ciphertext must remain below a certain level threshold. This error, added at the moment of the encryption, grows with the number of operations. At some point, the noise level will become too large and it will be impossible to decrypt correctly. There are Somewhat Homomorphic Encryption (SHE) schemes for which the multiplicative-depth (or, rather, level of noise tolerance) has to be specified at setup time. . This means that for a given depth L, an encryption scheme can be parameterized to compute L operations correctly. A scheme allowing for unbounded computations can be parametrized a priori, with a set of parameters that would fit to be tailored for a specific application over any computation.

Functional Encryption: It was mentioned for the first time in the work of Adi Shamir [25] as Identity-Based Encryption (IBE) without introducing the name itself. But it took the functional encryption name for the first time with the work of Dan Boneh et al. [26] in 2010. It was originally aimed to generalize the description of access control in public-key encryption, gathering together protocols like Identity-Based Encryption, Inner Product Encryption, or Broadcast Encryption. In detail, it is an asymmetric cryptographic scheme, that enables a key holder to learn a specific function on encrypted data, without learning anything else about the data. It can be a good candidate for certain classes of machine learning applications such as confidential machine learning. The work [4, 27, 28] used the FE in the confidentiality-preserving machine learning. Therefore, the FE technique seems a solution to solve the problem of confidentiality when a server needs to compute a function over encrypted data, without learning anything else about this data. One of

the drawbacks of FE solutions is the need to introduce an authority. This entity holding a master secret key msk can generate a key sk_f that enables the computation of the function f on encrypted data and a public key pk. Now, the client uses the mpk to encrypt the data and send the encrypted form to the server. The authority will derive a secret key sk_f from its msk and send this key to the server. So the server can calculate the function f using this key sk_f, over the client's data and obtain a clear result (i.e. plaintext result). Currently, to the best of our knowledge, at the time of writing this thesis, there are two propositions of FE: one can evaluate scalar product e.g. [29], and the other can evaluate degree-two polynomials [30]. Briefly, FE evaluates and decrypts over encrypted data versus FHE evaluates over encrypted data.

2.4.2 Integrity-Preserving Tools

Integrity can be ensured in different ways, but in this work, we focus on the cryptographic tools (without hardware solutions). There are several approaches in which integrity can be ensured when outsourcing some type of machine learning computations. We present them in the following:

Hash Function: Introduced in 1953 by German inventor Hans Luh [31] the hash function that maps any message of a variable length into a small digest of fixed length with a strong collision resistance. In general, hash functions are divided into two large categories: cryptographic hash function (like MD5 [32], SHA−3 [33], etc) or non-cryptographic hash function (checksum function[34], Cyclic Redudaucy Checks CRC [35]). The first designs of cryptographic hash function date back to the 1970s, where their importance was first realized with the invention of Diffie and Hellman cryptosystem [36]. It grew quickly, and more schemes were proposed in the 1980s. In recent years, there were some attacks on hash functions as for example, complete collisions of SHA-0 [37, 38], and a work of Wang Xiaoyun announced a collision, including MD4 [39], MD5 [40],etc. In recent years new secure hash functions have been proposed by NIST as SHA−2 [41], SHA−3 [33]. In general, hash functions are used as a tool to achieve the integrity of data. **Zero-Knowledge Proof (ZKP)**: ZKP methods are recent and were first developed in the late 80s, where it appears in [42] written by: Shafi Goldwasser, Silvio Micali, and Charles Rackoff. It has become a subject of study in cryptography. The ZKP constructs to prove an argument while yielding nothing beyond its truth. More precisely, it enables an entity called prover to convince another entity called verifier of an assertion without revealing more information than strictly necessary to convince her. Numerous works proposed in order to improve the efficiency of zero-knowledge proof [43–48]. The work of Ishai et al. [49] use the MPC as a building block in the design of efficient and conceptually simple zero-knowledge proofs.

Verifiable Computing: VC aims at verifying the remote execution of a function on an untrusted machine. Namely, it enables a client to delegate to another entity (in most cases a server) the computation of a function. The other entity evaluates the function and returns the result with a proof that the computation of the function was carried out correctly. This property has been extensively studied in the literature [50–55]. In the context of Verifiable computing over encrypted data it is addressed in the seminal paper of Gennaro et al. [56], where they introduced the notions of non-interactive verifiable computation, and they build a VC scheme for arbitrary function using garbled circuits secrets and FHE. Goldwasser et al. [57] demonstrated how they can use their

succinct single-key functional encryption scheme in order to build a VC protocol. In spite of advances in both of these two solutions in order to ensure both confidentiality and integrity, they are still unpractical, e.g. [56] require the full FHE power, and [57] required attribute-based encryption for expressive predicates and work for functions with single-bit outputs. Stimulated by the growing need for verifiable computing on encrypted data for general-purpose computation, in 2014 Fiore et al. [3] presented an efficient VC scheme on encrypted data to delegate: polynomials of large degree, linear combinations, linear functions over the rings $\mathbb{Z}_{2^k}$ and multivariate polynomials of degree 2. In 2020 Fiore et al. presented new work in this line of research [58] allowing the computation of multivariate polynomial degree ensuring public verifiability. However, they miss an implementation, and it works with particular parameters (q prime larger than 2^λ) for the homomorphic parameters.

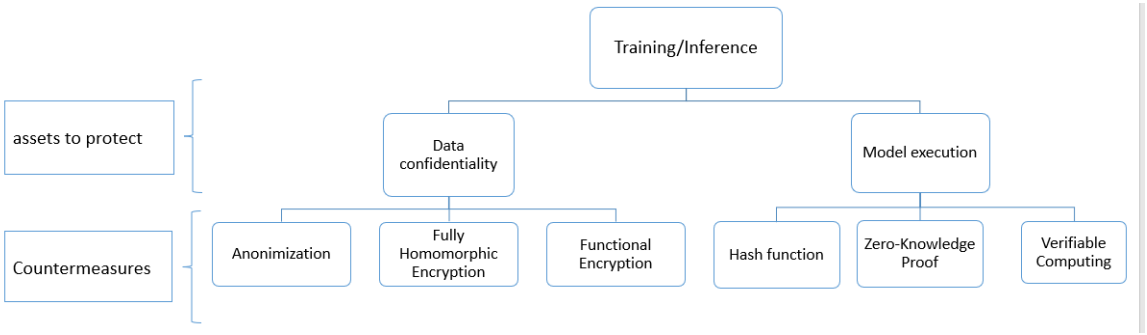

Figure 2.3: This figure presents the assets to protect and the countermeasure that we have in both phases of ML.

Figure 2.3 summarizes the threats against the ML, and more precisely, against the data and the model in the training and the inferences phases, with the corresponding countermeasures. This thesis aims to ensure an outsourced secure ML evaluation by ensuring the data confidentiality and the execution integrity for training or inference algorithms by means of FHE and VC schemes.

2.5 Use Case

To clarify our context, we present the following ML applications using FHE and VC. The first application permits to a hospital to create a secure training collaboration with other healthcare service provides, while avoiding disclosure of the local hospital training data. By means of FHE and VC we ensure the model confidentiality and its integrity with regards to the threat coming from the central server making the global model update.

The second application aims to ensure a secure prediction over an external malicious server which evaluates the model over FHE encrypted user's data while ensuring the integrity of model execution using VC scheme.

2.5.1 Machine Learning Training Application

For the training, we choose to focus ourselves on the Federated Learning framework which allows several healthcare providers to collaboratively train a common AI model, with the use of a central server. Each health service provider evaluates a local training algorithm

over clear patients' data (like CT-scan, patients' folder, analysis results), encrypts the result of this algorithm and generates associated tags forwarded to the central server as in Figure 2.4. The central server runs an averaging of the model parameters in the encrypted domain to update the global model. At the same time, the server runs the averaging over the input tags to generate an integrity tag associated with the averaged model parameters. The server sends back the encrypted results with the tag to each participant. They check the calculation of averaging and if it is ok, they decrypt the result to obtain the updated global model. Hence, each local trainer is ready for a new iteration of the protocol.

For the threats: we will guarantee the confidentiality of the local model for a given participant against the server threat by means the homomorphic encryption. Further, we will achieve the integrity of the central server calculation by means of the VC scheme.

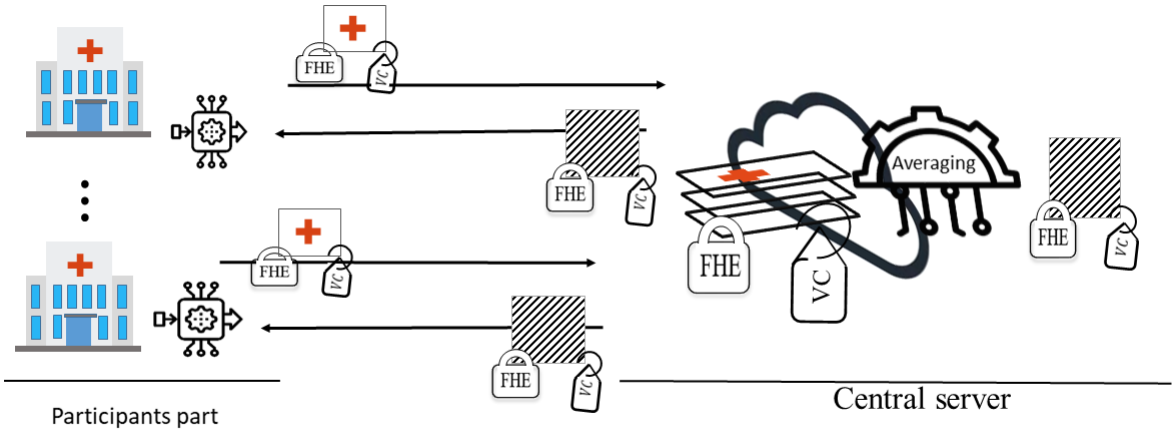

Figure 2.4: A figure presenting the training application between two types of entities, the participants in the learning phase and the server that updates the training model.

2.5.2 Machine Learning Inference Application

Here, a user can be diagnosed based on his/her CT-scan using a secure inference model. Firstly, like in Figure 2.5 the CT-scan is encrypted and an associated integrity tag is generated. The patient encrypted data as well as the integrity tag are sent to the inference provider server (that can be a malicious server), which evaluates the inference model over encrypted data and over the integrity tag and sends back the encrypted diagnosis result with the integrity computation tag. Finally, the practitioner of patient checks that the inference model was computed correctly, and if so, he will decrypt the received encrypted result. Otherwise, the clients will take appropriate actions depending on the protocol used. Therefore, the confidentiality of the CT scan of a given user is achieved against the threats coming from both the inference provider using homomorphic encryption, and the integrity of the model is preserved by means the verifiable computing against the integrity threat coming from a malicious server (the inference provider).

This chapter presents the ML with the associated threats coming from different entities servers or adversaries, and we summarize the tools used against these threats. In the following chapter, we detail the first tool, *Homomrphic Encryption*, used in our contribution.

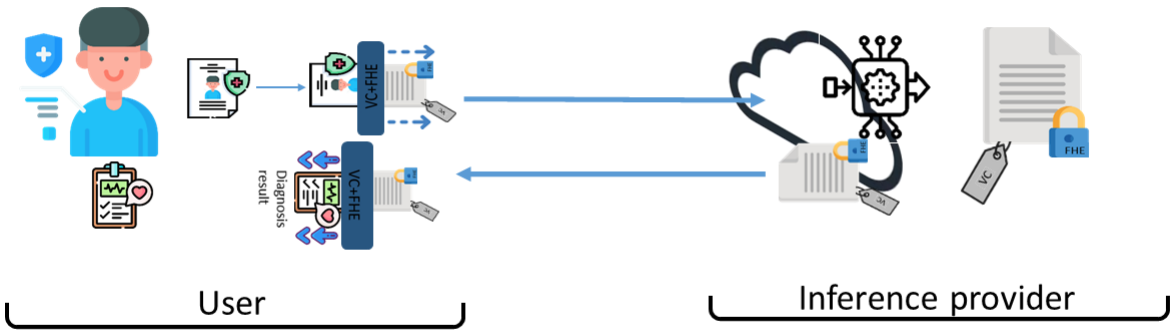

Figure 2.5: A figure presenting the inference application between two entities, the participants in the inference phase and the server that provide the ML algorithm. When the user receive the encrypted result, it checks that the model was computed correctly using VC and if so, it decrypts the result using FHE to obtain the diagnosis result.

Chapter 3

Homomorphic Encryption

A homomorphic cryptosystem is a special type of encryption allowing to perform computation directly over encrypted data. A schema is said to be fully homomorphic if it allows to perform both addition and multiplication over encrypted data. Finding a fully homomorphic cryptosystem remained an open problem until the beginning of the second millennium, more precisely until 2009 with the outbreak of Gentry scheme [19]. Even if there existed already partially homomorphic schemes such as multiplicative homomorphic encryption like RSA[18] or the additive homomorphic schemes as Paillier cryptosystem [22] (allowing only one type of operation), they were not expressive enough to be used in real world applications.

In abstract algebra, the homomorphism is seen as a map, that transforms all the algebraic structures (like operation) of one domain/algebraic set to another domain/algebraic set. This is the same idea for homomorphic encryption in the cryptographic domain, where the homomorphic encryption allows to convert the addition or multiplication operations between the plaintext and ciphertext spaces. This means that one can apply an addition or multiplication over encrypted data without decrypting and the result corresponds to the application of this operation over the corresponding plaintext. Thus, the homomorphic encryption can be seen as a cryptosystem that converts the operation from the encrypted layer to the clear layer while preserving the confidentiality of data.

3.1 Security

3.1.1 Security Notions

One approach for proving the security of a cryptographic scheme is called provable secure, which divides in classes the type of security of the scheme depending on the attacker's capabilities and security goals that are needed. In this context, there are four classes of security for public-key cryptosystem: perfect secrecy, semantic security, indistinguishability, and Non-malleability security.

- *Perfect secrecy*: It was proposed in the 19th century, by the Netherlands cryptographer Auguste Kerckhoffs and it was reformulated by the American mathematician Claude Shannon [59]. It states that that the security of an encryption system must reside only in the key and not in the cryptosystem, supposing that the attacker has infinite resources and time.

- *Semantic security (SS)*: SS was introduced by Goldwasser and Micali [60] as the following notion: with a given ciphertext an adversary cannot obtain any partial information about the plaintext. In this context the adversary is supposed to behave as a probabilistic polynomial-time Turing machine.

- *Indistinguishability (IND)* [61]: If a cryptosystem possesses the property of indistinguishability, then an adversary with polynomial bounded computational resources will be unable to distinguish pairs of ciphertexts based on the message they encrypt.

- *Non-malleability (NM)* [62]: Malleability is the ability for an encryption algorithm to transform a ciphertext into another ciphertext which decrypts to a related plaintext. In other words, if an encryption scheme is Non-Malleable, then an adversary cannot generate a ciphertext from a different ciphertext i.e. for a given ciphertext $Enc(x)$, an adversary cannot generate another ciphertext that decrypts $f(x)$ for a known function f, without necessarily knowing x.

In the context of cryptanalysis, there are several attacker models depending on the adversary access to the cryptosystem:

1. *Chosen Plaintext Attacks (CPA)*: An adversary knows the plaintext and the corresponding ciphertext. In this model, we say that adversary has the access to the encryption oracle.

2. *Chosen Ciphertext Attacks (CCA1)*: in addition to the CPA, the attacker has the access to a decryption oracle before it obtains a challenge ciphertext. This means that the attacker can obtain the descriptions of chosen ciphertexts.

3. *Adaptive Chosen Ciphertext Attacks (CCA2)*: the adversary has access to the decryption/and encryption oracle even after it obtains a challenge ciphertext. The non-degeneracy condition is that the adversary cannot use this access to decrypt the challenge itself.

The combination of these security classes and these attacker methods produces security levels for semantic and malleability security like IND-CPA, IND-CCA1, and so on. For a given cryptographic scheme, the IND-CPA, IND-CCA1, IND-CCA2 are formalized as a game between an adversary and some honest challenger. We say that this scheme is

semantic secure in the sense of Indistinguishability under Chosen Plaintext Attack (IND-CPA) if an efficient adversary cannot win the IND-CPA game.

1. First, the challenger publishes the public key that corresponds to the cryptographic scheme of the object of study.
2. Second, the adversary selects pairs of plaintext message and send them to the challenger.
3. Next, the challenger sends an encryption of only one of the plaintexts (at random) back to the adversary.
4. Finally, the adversary determines which one of the plaintext messages was encrypted.

We say that the adversary wins if it succeeds in the guess of plaintext according to the ciphertext sent by the challenger.

The IND-CCA1 is the same as the IND-CPA, but with the variation in the step 3 of the game, where the adversary has access to decryption oracle before the challenge ciphertext is sent. This means she can decrypt arbitrary messages at will, and even after seeing the target ciphertext in the IND-CCA2. Actually, these notions are related among them [63], and their relations are described in the Figure 3.1, where the CCA2 is stronger than CCA1 and also than CPA with respect to the attacker models. Regarding the goals, the NM model implies IND in general, but, in the CCA2 model, IND also implies NM, and the IND is equivalent with the SS.

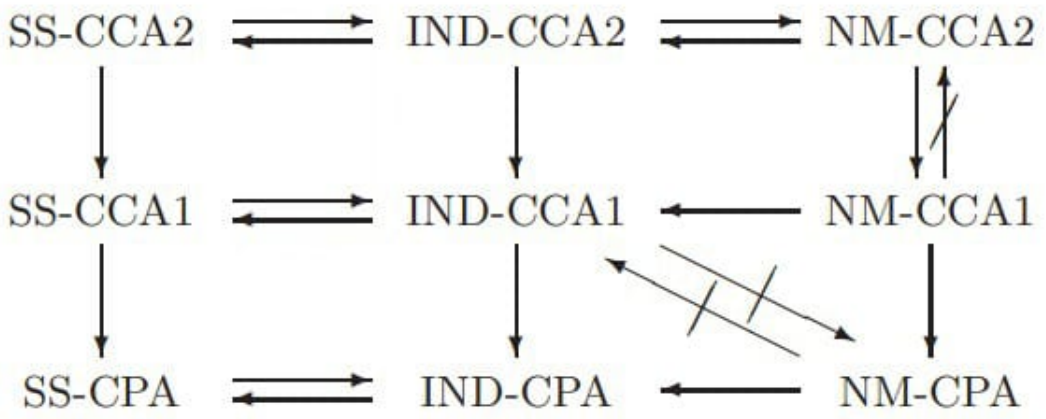

Figure 3.1: Relations between security notions [63]

Indeed, we note that the time taken by the adversary to win the game defines the security of the cryptographic scheme. In detail, we call λ the security parameter of a scheme, with λ a given integer.The adversary must not be able to win the game in $O(2^{\lambda})$ operations.

3.1.2 FHE Security

Since they allow computations directly over encrypted data, it is simple to observe that all FHE schemes are malleable by construction because any scheme that supports homomorphic operations is malleable. Then it remains, to define the position of the homo-

morphic encryption scheme against the three security notions IND-CPA, IND-CCA1 or IND-CCA2.

The best we can get for homomorphic encryption schemes is IND-CCA1 because it is not hard to see that homomorphism contradicts IND-CCA2 security, since, in the IND-CCA2 the adversary can ask to encrypt m by the encryption oracle. Next, the adversary adds the output ciphertext to the target ciphertext, then submits the newly resulting ciphertext to the decryption oracle to decrypt. It subtracts m to get the desired plaintext and wins the game without violating the conditions of the IND-CCA2 game (where the adversary can decrypt any text except the challenge ciphertext). Consequently, the adversary can determine which plaintext is hidden in the challenge ciphertext and win the game IND-CCA2.

For IND-CCA1 security, any scheme using Gentry's bootstrapping idea [19], cannot be IND-CCA1 secure by construction, since the bootstrapping idea is based on the publication of a bootstrapping key which is the encryption of the secret key that can be found in the public key. This is the weakness of the scheme using this idea, since, an adversary can simply ask to decrypt this key, that gives data of the entire secret key. On the other hand, other homomorphic encryption schemes are proven as indistinguishable under a non-adaptive chosen ciphertext attack (IND-CCA1) like [64, 65]. we use the BFV and Paillier schemes which are respectively IND-CPA and IND-CCA1 [66].

We will recall the notions of security used for any cryptographic scheme used in this thesis when they will be presented in the corresponding section.

3.1.3 Hardness Assumptions

The evaluation of cryptographic security proof, in other words, the time it takes for an efficient adversary to win the game, is a reduction from the adversary existence that violates the security notions to the intractability/hardness to solve one or more mathematical problems where it must be hard (hard in the means of resolution time) to solve it. These are known as computational hardness assumptions as for instance, the Integer factorization (i.e. factoring large composite numbers) assumptions used in RSA cryptosystem. When the hardness assumption has been defined, an adversary is supposed to use the fastest possible way to find a solution, therefore, win the game. Unmistakably, a cryptographer's goal is to create a cryptosystem that uses a "hard" problem[1], which makes solving it to require an outside time of nature (i.e. impractical time).

3.2 Brief History

In this section, we provide a slight overview of the development in the FHE models, not only for the purpose to recognize the previous works but also to determine the limit of this tool (i.e. how many operations can evaluate in encrypted domain), and thus define the limits of our ambition.

Before this, we will present briefly the FHE model from a theoretical point of view: as we presented above Homomorphic Encryption (HE) schemes allow to perform computations

[1]To demonstrate that a particular problem is "hard" is truly another difficult problem, and we cannot deep delve into this proof. Since it is out of our scope in this thesis, we will only refer to the reference of the proof when we use it.

directly over encrypted data without decrypting it first. That is, with a Fully homomorphic Encryption scheme E, we can compute $\mathsf{E}(\mathsf{m}_1 + \mathsf{m}_2)$ and $\mathsf{E}(\mathsf{m}_1 \times \mathsf{m}_2)$ from Encrypted messages $\mathsf{E}(\mathsf{m}_1)$ and $\mathsf{E}(\mathsf{m}_2)$.

3.2.1 Pre-FHE

In the first 30 years after the appearance of the first FHE notions [17], the field has slightly progressed. It started with the bit-wise additive Homomorphic encryption scheme proposed by Goldwasser and Micali in 1982 [60], which is the first probabilistic public-key encryption[2] scheme provably secure. In the same line of research, Pascal Paillier [22] invented an additive homomorphic encryption scheme which provides IND-CPA security. A few years later, specifically in 2005 Boneh et al. [67] have also designed a system of provable security encryption, which can evaluate quadratic multivariate polynomials on ciphertexts domain, i.e. can perform an unlimited number of additions, but just one multiplication in the ciphertext domain.

3.2.2 FHE-Generation

The turning point in the context of FHE schemes is the Gentry scheme [19, 20], with his ground-breaking bootstrapping idea, and to this day, several homomorphic encryption schemes proposed follow his blueprint. All the ciphertexts of the homomorphic encryption schemes are noisy in some sense, and this noise grows as one adds and multiplies ciphertexts until ultimately we can no longer decrypt. The main differences between all the homomorphic schemes concern the noise management technique, the evaluation function and the mathematical concepts that define them. Consequently, we can group them into three generations that will be presented in details in the following. We will summarize the main characteristics of each generation in Table 3.1.

1. First Generation: There are public-key encryption schemes based on the ideal lattice on the polynomial ring [19, 20] and using the Gentry's blueprint for the "bootstrapping" concept. This type of scheme supports both homomorphic addition and multiplication operations on the ciphertexts, where these operations are just an addition or multiplication over the polynomial rings. The Gentry's "noisy" bootstrapping represents a tool to diminish the growth of ciphertexts: it is a refreshing of an encrypted message using encryption of secret key and applying the decryption procedure homomorphically. This resets the noise and allows for more operations to take place. The resolution of the noise hurdle opens the door to more efficient and practical FHE. The problem is still the time efficiency. To reduce the decryption complexity, a squashing technique is proposed under the hardness assumption of the sparse subset-sum problem (SSS). The first implementation of this generation [68] achieved only a leveled homomorphic encryption (LHE) because it did not succeed in implementing the squashing. Later implementation of Gentry scheme [69] showed that a single bootstrapping operation takes between 30 seconds to 30 minutes depending on parameters.

2. Second Generation: It was born in 2011-2012 with Zvika Brakerski and, Vinod Vaikuntanathan with their BGV scheme [70, 71]. They built an efficient LHE scheme

[2] Probabilistic public-key encryption is a public-key encryption scheme where the ciphertext of the same message under the same public key differs on every run of the encryption algorithm.

based on the hardness of LWE or (Ring) Learning With Errors (RLWE) problem (see section 3.3 for definition). They introduced a noise-management technique called re-linearization to obtain a Somewhat Homomorphic Encryption (SHE) and replace the bootstrapping procedure by the operations of Key switching and modulus switching to transform it into LHE, where the homomorphic evaluation and decryption climb a ladder of decreasing modulus and scale the ciphertext properly to ensure correctness. This technique improves dramatically the performance by resulting in a slower noise growth during homomorphic computation. The BGV is a reference scheme of this generation. Brakerski et al. [72] simplified the BGV construction and improved the underlying assumptions using the hardness of classical GapSVP. Fan et al. [24] improved the last scheme by adapting it to Ring-LWE setting. All these schemes have a seemingly complex multiplication.

3. Third Generation: The third generation of FHE saw the light with the works of Gentry et al. in 2013 where they produced a new scheme GSW [73] in order to avoid the BGV-like encryption drawback and the re-linearization step of the second generation schemes. The authors of [74] proposed a new bootstrapping method to improve the bootstrapping of [75]. By eluding the inefficiencies resulting from the use of Boolean circuits and Barrington's theorem used in [75], the number of homomorphic operations on GSW ciphertexts is optimized compared with the one from [75]. Hence, both FHEW [76] and TFHE [77, 78] obtain a more efficient bootstrapping operation.

1^{st} generation Gentry09	2^{nd} generation BV11	3^{rd} generation GSW13
Not highly efficient ideal lattice	Much more efficient (R) LWE assumption No known weaknesses	Generally less efficient Safety slightly better assumption

Table 3.1: The main features of each generation

3.3 Technical Preliminaries

3.3.1 General notions

Sets. Given two real numbers a,b such that $a \leq b$, [a,b] designates the set of all real numbers between (and including) a and b.

$\mathbb{Z}/q\mathbb{Z}$ denotes the set of integers modulo q with a given integer q, i.e. it is a set in $[0, q-1]$. We denote by $\mathbb{Z}_q$ the set $(-q/2, q/2]$, as such it should not be confused with the above set.

Given $x \in \mathbb{R}$, $\lfloor x \rceil$ is the rounding to the nearest integer and $\lfloor x \rfloor$, $\lceil x \rceil$ to indicate rounding up or down.

We write the vector a as $\vec{a}$. Every value is clearly presented as a vector or scalar value.

Modulus. Given a positive integer q, $r_q(t)$ denoted the remainder modulo q (i.e. t *modulo* q) into $[0, q)$. Then $c = \Delta \cdot q + r_q(c)$, where $\Delta = \lfloor c/q \rfloor$.

$[\cdot]_q$ denotes the reduction modulo q into the interval $(-q/2, q/2]$ of any integer or any polynomial integer (obtained by applying $[\cdot]_q$ to all its coefficients).

Polynomial Ring. The ring $\mathbb{Z}[x]/(f(x))$ is denoted by $\mathcal{R}$, where $f(x) \in \mathbb{Z}[x]$ is a monic irreducible polynomial of degree d.

Given an integer q, we define the $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ as a set of polynomials in $\mathcal{R}$ with coefficients in $\mathbb{Z}_q$. The polynomial $a \in \mathcal{R}$, will be denoted by $a = \sum_{i=0}^{d-1} a_i \cdot x^i$ and a can be presented as the vector $\vec{a} = (a_0, \ldots, a_{d-1})$.

For a fixed integer w and $l_{w,q} = \lfloor log_w(q) \rfloor + 1$, a polynomial $a \in \mathcal{R}_q$ can be written in base w as $\sum_{i=0}^{l_{w,q}-1} a_i \cdot w^i$.

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

With $\zeta_n = e^{2\pi i/n}$, the $n-$ th cyclotomic polynomial is defined as:

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

Dot product. The dot product of two vectors $\vec{u} = (u_0, \ldots, u_{d-1})$, and $\vec{v} = (v_0, \ldots, v_{d-1})$ is defined as:

$$\vec{u} \cdot \vec{v} = < u, v > := \sum_{i=0}^{i=d-1} u_i v_i$$

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

Gaussian distribution :

The Gaussian distribution over $\mathbb{R}$ centered at μ (i.e. the expectation) with a standard deviation $\sigma \in \mathcal{R}^+$ is denoted by $\mathcal{N}(\mu, \sigma)$ and defined with the following density function:

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}.$$

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

The **discrete Gaussian** distribution over $\mathbb{Z}$ centered on 0, with standard deviation σ, denoted $\mathcal{D}_{\mathbb{Z},\sigma}$, is the probability distribution that assigns a probability proportional to $e^{\frac{-\pi|x|^2}{\sigma^2}}$ to each $x \in \mathbb{Z}$.

Definition 1. (WordDecomp & PowersOf) For any $a \in \mathcal{R}$ and $b \in \mathcal{R}$, with coefficients in $(-w/2, w/2]$, we define **WordDecomp** and **PowersOf** as follows:

$$\begin{cases} WordDecomp_{w,q}(a) = ([a_0]_w, \ldots, [a_{l_{w,q}-1}]_w) \in R^{l_{w,q}}. \\ PowersOf_{w,q}(b) = ([b \cdot w^0]_q, \ldots, [b \cdot w^{l_{w,q}-1}]_q) \in R^{l_{w,q}}. \end{cases}$$

With this definition, we can observe that:

$$\langle WordDecomp_{w,q}(a), Powersof_{w,q}(b) \rangle = ab \bmod q$$

Distribution probability. We note by χ_{key} and χ_{err} two discrete, bounded probability distributions on $\mathcal{R}$ defined as follow:

$$\begin{cases} \chi_{err} : \text{ discrete Gaussian distribution with parameter } \sigma. \\ \chi_{key} : \text{ distribution in } (\{-1, 0, 1\}), \text{ s.t.} \\ Pr([x = -1]) = Pr([x = 1]) = 1/4 \text{ and } Pr([x = 0]) = 1/2. \end{cases}$$

3.3.2 Learning With Error

The LWE problem. The LWE problem was introduced in 2005 by Regev [79]. Informally, the LWE problem consists in solving an over determined, but noisy linear system, modulo an integer q. The LWE problem is defined around three parameters: the dimension n, the modulus q and the error factor α. The parameters q and α are chosen according to n. We distinguish the two versions of the LWE problem; the *Computational* and the *decisional* LWE versions.

Definition 2 (**LWE Distribution**). We let $n \geq 1$, $q \geq 2$ be two integers, and let χ be a fixed noise probability distribution over $\mathbb{Z}$. Let $\vec{s} \in \mathbb{Z}_q^n$ be a secret vector called the secret. The *LWE distribution* denoted by $\mathcal{A}_{\vec{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is obtained as follows.

1. Sample a vector $\vec{a} \xleftarrow{\$} \mathbb{Z}_q^n$,
2. Choose $e \xleftarrow{\chi} \mathbb{Z}$,
3. Evaluate $b =< \vec{a}, \vec{s} > +e \bmod \text{q} \ \in \mathbb{Z}_q$,
4. Output $(\vec{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

Definition 3 (**search-LWE**). The problem is to find $\vec{s} \in \mathbb{Z}_q^n$, from a given arbitrary n samples

$(\vec{a}, b) \xleftarrow{\mathcal{A}_{\vec{s},\chi}} \mathbb{Z}_q^{n+1}$.

Definition 4 (**Decision-LWE**). It is the problem to distinguish the LWE distributions $(\mathcal{A}_{s,\chi})$ from uniformly random samples of $\mathbb{Z}_q^n \times \mathbb{Z}_q$, i.e. determining whether a given $(\vec{a}, b)$ was taken form $\mathbb{Z}_q^{n+1}$ according to $\mathcal{A}_{s,\chi}$ or were generated uniformly at random.

These two problems reduce to each other in polynomial time, and LWE is proved as a hard problem. Regev [79] proposed a public-key cryptosystem based on the hardness of LWE.

In general, the distribution χ for LWE distribution is considered to be a discreet Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma}$, where $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$ with a given α.

The Ring-LWE problem. It first defined in the paper of Lyubashevsky et al. [80], in which they introduced the algebraic variant of LWE called *Ring-LWE*. We will see in the following that the structure of RLWE is very similar to LWE but over polynomial rings.

RLWE is parametrized by a polynomial ring $\mathcal{R}$, a modulus $q \geq 2$ defining the quotient ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$, and a noise probability distribution χ over $\mathcal{R}$.

Specifically, we take $\mathcal{R} = \mathbb{Z}[x]/(f(x))$ to be a cyclotomic ring with cyclotomic polynomial f of degree d, and χ is discretized Gaussian in the canonical embedding of $\mathcal{R}$.

Definition 5. **RLWE-Distribution**. For an $s \in \mathcal{R}$, called the secret, the *RLWE-Distribution* $\mathcal{A}_{s,\chi}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is sampled by:

1. Choose $a \xleftarrow{\$} \mathcal{R}_q$,
2. Select $e \xleftarrow{\chi} R$,
3. Evaluate $b = a \cdot s + e \mod q$,
4. outputting $(a, b) \in \mathcal{R}_q \times \mathcal{R}_q$.

Definition 6 (**Search- RLWE**). It is the problem to find $s \in \mathcal{R}_q$, from a given arbitrary samples $(a, b) \xleftarrow{\mathcal{A}_{s,\chi}} \mathcal{R}_q^2$.

Definition 7 (**Decision-RLWE**). It is the problem of distinguishing the RLWE distributions $(\mathcal{A}_{s,\chi})$ from uniformly random samples of $\mathcal{R}_q \times \mathcal{R}_q$, i.e. determining whether a given (a, b) was taken form $\mathcal{R}_q \times \mathcal{R}_q$ according to $\mathcal{A}_{s,\chi}$ or were generated uniformly at random.

Same as for LWE problem, the Decision-RLWE and Search-RLWE are equivalent problems.

Hardness of (R)LWE-problem To prove the difficulty of these two problems, a reduction was used to show that an LWE problem can be reduced to the shortest vector problem (SVP) over ideal lattices (using a classical reduction [81, 82] or using quantum algorithm [79, 83]). Then, to solve the LWE problem is at least difficult as to solve all instances of a variant of the SVP problem. For the RLWE problem, it has been proved difficult, under certain restrictions, by reductions to a variant of SVP [84]. As shown in [80, 85] s can be sampled from χ instead of being taken uniformly in $\mathcal{R}_q$ without any security implications. Moreover, the hardness of this problem is independent of the precise shape of q [86]. As such q does not have to be prime and can be taken simply as a power of 2.

(R)LWE encryption cryptosystem Generally, all the (R)LWE schemes are based on the adding error principle, i.e. the encryption procedure consists in adding an error or noise into ciphertext which must not exceed a certain threshold to ensure decryption. This noise increases as the operation progresses (like addition or multiplication) inducing an error propagation. We note that the RLWE encryption is more efficient than the LWE peer due to its compactness (each b is N-dimensional) and due to implementation and optimizations, such as the use of a Fast-Fourier-Transform.

Here, we present a basic (R)LWE-based encryption scheme with no homomorphic operations. Next in the following sections we present BGV and BFV encryption scheme. For simplicity, we present the (R)LWE-based encryption scheme here with the plaintext space taken as $\mathcal{R}_2$. It is easy to generalize it to work with larger plaintext $\mathcal{R}_t$ for some integer $t > 1$.

Basic (R)LWE-Based Encryption Scheme (E):

E.ParamGen(λ, μ, b): Use the bit $b \in \{0, 1\}$ to determine whether we are setting parameters for a LWE-based scheme (where $d = 1$) or a RLWE-based scheme (where $n = 1$). Choose: a $\mu-$bit modulus q $(\mu = log_2(q)), d = d(\lambda, \mu), n = n(\lambda, \mu), N = \lceil (2n+1)log(q) \rceil$, the distribution χ appropriately to ensure that this scheme achieves 2^λ security against known attacks. Let $params = (q, d, n, N, \chi)$.

E.SecretKeyGen(params): Sample $\vec{s'} \xleftarrow{\chi} \mathcal{R}_q^n$. Set $sk = \vec{s} \leftarrow (1, s'_1, \ldots, s'_n) \in \mathcal{R}_q^{n+1}$.

E.PublicKeyGen$_{sk}$*(params)*: Takes as input a secret key $sk = \vec{s} = (1, s')$ and $params$. It generates the matrix $A' \leftarrow \mathcal{R}_q^{N \times n}$ uniformly and samples a vector $\vec{e} \xleftarrow{\chi} \mathcal{R}^N$. It computes $b = A'\vec{s'} + 2e$. and sets A to be the $(n+1)-$column matrix consisting of b elements followed by the columns of the matrix $-A'$, Namely $A = (\underbrace{A's' + 2e}_{n+1} | \underbrace{-A'}_{n+1})$. Finally the output is $pk = A$.

E.Enc$_{pk,params}(m)$: To encrypt a message $m \in \mathcal{R}_2$, set $m = (m, 0, \ldots, 0) \in \mathcal{R}_q^{n+1}$, sample $r \leftarrow R_2^N$ and return the ciphertext $c = m + A^T r \in \mathcal{R}_q^{n+1}$.

E.Dec$_{sk,params}(c)$: Compute $m = \Big[[< c, s >]_q \Big]_2$.

The correctness is easy to prove. For security, the above scheme is based on the LWE assumption for $d = 1$ or RLWE assumption for $n = 1$. The above (R)LWE-based cryptosystem can be proven to be semantically secure assuming the hardness of (R)LWE given 3 samples [80]. We note that to achieve 2^λ security against known lattice attacks, one must have $n \cdot d = \Omega(\lambda \cdot log(q/B))$ where B is a bound on the length of the noise (i.e. $||\chi|| < B$), see e.g. [80].

LWE-based cryptosystem security

The security of LWE-based encryption is evolving over time because it depends immediately on the attack strength against them. The detection of faster LWE-attacks usually means a modification of parameters used on LWE-based schemes is necessary, at a performance cost. These parameters are set according to a desired security level λ defined in chapter 3. One of the strengths of this scheme is the easy modification of their parameters.

Two important works studied the hardness of the LWE problem [87, 88]. The oldest among them [87] provides a software tool called LWE-estimator as a Sage module to estimate the hardness of concrete LWE instances. More precisely, it enables the users to estimate the running times of the various attack algorithms for particular parameter choices. That makes the selection of parameters for lattice-based primitives much easier and more comparable. Also, it collects the existing attacks on LWE and affords a minimal-security parameter λ for a clear estimation of the security of the scheme implemented with those parameters. An up-to-date of these results is presented in [88]. This estimator is kept

up-to-date with the latest advancement in the field of cryptanalysis. Then this is the best source of security required to select the parameters of a LWE-based cryptosystem.

This estimator supposed that it has access to an optimal number of samples $(\vec{a}, b)$ (according to the attacker) to solve the LWE problem. Bindel et al. [89] analyze the hardness of LWE instances considering a limited number of samples, based on the LWE estimator. This can provide a more practical choice of parameters for LWE-based schemes, from a security point of view.

RLWE-based cryptosystem security It is generally supposed that any RLWE instance, can be reduced to its equivalent version of the LWE instance. As long as it is true, the security of RLWE can be considered equivalent to the security of LWE, and this is the case at the time where we are writing this thesis. The problem of RLWE is still widely used in the cryptography world due to its great services like speedy operations and shorter keys.

All of the LWE-based encryption schemes used in this thesis were parametrized with the LWE estimator, under the unlimited access of the attacker to the samples. The cryptanalysis question of LWE-based cryptosystem is out of the extent in this work.

3.3.3 HE schemes

3.3.3.1 Paillier cryptosystem

As its name indicates, the cryptosystem of Paillier was invented by Pascal Paillier in 1990 [22]. It is an additive homomorphic cryptosystem, based on the hardness of computing the n-th residue classes, which is believed to be computationally difficult. In the following, we recall the general principles of this cryptosystem.

KeyGen$(sz) \rightarrow (pk, sk)$: It generates the keys for the cryptosystem taking as input the number of bits sz of the modulus.

Choose two large prime numbers p_E and q_E such that $\lambda = \text{lcm}(p_E - 1, q_E - 1)$, and set $N_E = p_E * q_E$. We note that the cleartext domain is $\mathbb{Z}_{N_E}$ and the ciphertext domain is $\mathbb{Z}_{N_E^2}$.

Select a random $g < N_E^2$ such that $\gcd(L(g^\lambda \mod N_E^2), N_E) = 1$,with $L(u) = \frac{u-1}{N_E}$.

Set $pk = (N_E, g)$ and $sk = (p_E, q_E)$.

Enc$_{pk}(m) \rightarrow c$: The encryption algorithm produces a ciphertext c using the public key pk by computing $c = g^m r^{N_E} \mod N_E^2$, where $m < N_E$ is the message and r is uniformly chosen in $\mathbb{Z}_{N_E}$.

Dec$_{sk}(c) \rightarrow m$: The decryption algorithm is made by computing the plaintext m from the ciphertext c, using the private key sk as follow:

Letting $D = L(g^\lambda \mod N_E^2)$ and D^{-1} its multiplicative inverse in $\mathbb{Z}_{N_E}$, the decryption is performed by evaluating

$$m = \text{Dec}(c) = L(c^\lambda \mod N_E^2) \times D^{-1} \mod N_E.$$

More importantly, for the present purpose, this cryptosystem has the following homomorphic properties:

1. $\text{Dec}(\text{Enc}(m_1)\text{Enc}(m_2)) \mod N_E^2 = m_1 + m_2 \mod N_E$ (addition of two encrypted messages).
2. $\text{Dec}(\text{Enc}(m)g^k) \mod N_E^2 = m + k \mod N_E$, for all $k \in \mathbb{Z}_{N_E}$ (addition of an encrypted message to a clear integer).
3. $\text{Dec}(\text{Enc}(m)^k) \mod N_E^2 = km \mod N_E$, for all $k \in \mathbb{Z}_{N_E}$ (multiplication of an encrypted message by a clear integer).

Theorem 1. [22] The cryptosystem showed above does provide semantic security against chosen-plaintext attacks (IND-CPA) if and only if the Decisional Composite Residuosity Assumption holds.

3.3.3.2 BGV

The BGV homomorphic encryption scheme was proposed in 2011 by Brakerski, Gentry and Vaikuntanathan[3] [23], based on LWE and on the RLWE instances. The RLWE instance of BGV achieves a better performance than the LWE version. The idea is to use the modulo switching introduced in [90], in order to keep the error of the ciphertext under the limit that permits to decrypt it. This switching consists of a mapping of a ciphertext $c \in \mathcal{R}_q$, to a ring $\mathcal{R}_p$ where $q > p$, which still encrypts the same plaintext with keeping the error e contained within the ciphertext at the same level. This permits to multiply two ciphertexts and keep the error at the same level. One can apply this concept indefinitely, which opens the way for an FHE scheme. In the following, we will present in detail the outline of this scheme. Before this, we start by reminding the reader some definitions from [23]: *BitDecomp, PowersOf2, SwitchKeyGen, and Scale*, needed to explain the homomorphic encryption operations for this scheme.

(BitDecomp & PowersOf2) For any $\vec{a} \in \mathcal{R}_q^n$, and $l = \lfloor log\ q \rfloor$ we define **BitDecomp** and **PowersOf** as follows:

$$\begin{cases} BitDecomp(\vec{a}) = (a_{1,0}, \ldots, a_{1,l}, \ldots, a_{n,0}, \ldots, a_{n,l}) \in R_q^{l \cdot n}. \\ PowersOf2(\vec{a}) = (\vec{a}, 2 \cdot \vec{a}, \ldots, 2^l \cdot \vec{a}) \in R_q^{l \cdot n}. \end{cases}$$

where $a_{i,j}$ is the j-th bit in a_i's binary representation .

With this definition, we can observe that:

$$\langle BitDecomp(a), Powersof(b) \rangle = ab \bmod q$$

SwitchKeyGen: permits to obtain a new ciphertext c_2 of the same message of c_1, but under a secret key s_2. It proceed as follows:

SwitchKeyGen($s_1 \in \mathcal{R}_q^{n_1}, s_2 \in \mathcal{R}_q^{n_1}$):

1. Run $A = E.PublicKeyGen(s_2, N)$, for $N = n_1 \cdot \lceil log\ q \rceil$, where n_1 is the dimension of s_1.
2. Set $B = A + PowersOf2(s_1)$,
3. Output $\tau_{s_1 \to s_2} = B \in \mathcal{R}_q^{n_2}$, where n_2 is the dimension of s_2.

[3] last update in 2014.

SwitchKey$(\tau_{s_1 \to s_2}, c_1)$: Output $c_2 = BitDecomp(c_1)^T \cdot B \in \mathcal{R}_q^{n_2}$.

Scale$(\vec{x}, q, p, r)$: is defined as the operation taking as input the vector $\vec{x} \in \mathcal{R}$, the modulus p, q, and r and outputting x' the $\mathcal{R}-$vector closest to $(p/q) \cdot x$ that satisfies $x' = x \ mod \ r$.

Let us now describe the SHE scheme BGV as a 6-uplet (BGV.Setup, BGV.KeyGen, BGV.Enc, BGV.Dec, BGV.Add, BGV.Mult) as follows:

- **BGV.Setup**$(1^\lambda, 1^L)$: it takes as input the security parameter λ and the numbers of levels L of arithmetic circuit that we want the BGV be able to evaluate.

  1. Run $params_j \leftarrow E.ParamGen(\lambda, (j+1)\cdot\mu, b)$ to obtain a ladder of parameters, including a ladder of decreasing moduli from $q_L((L+1) \cdot bits)$ down to $q_0(\mu$ bits).
  2. Selects the discrete Gaussian distribution(denoted by χ_{err}) as the error distribution.
  3. Set params=($q_0, \ldots, q_{L-1}, \chi_{err}, L$), as public parameters.

- **BGV.KeyGen**(params):
  For $j = L$ to 0 do the following:

  1. Run s_j =E.SecretKeyGen($params_j$).
  2. Run A_j =E.PublicKeyGen($params_j, s_j$).
  3. Compute $s'_j = s_j \otimes s_j \in \mathcal{R}_{q_j}^{\binom{n_j+1}{2}}$, where the $\otimes$ denotes the vector tensoring operator.
  4. If $j \neq L$, run $\tau_{s'_{j+1} \to s_j} \leftarrow SwitchKeyGen(s'_{j+1}, s'_j)$.

  Put $sk = (s_0, \ldots, s_L)$.
  Set $pk = (A_0, \ldots, A_L, \tau_{s'_L \to s_{L-1}}, \ldots, \tau_{s'_1 \to s_0})$.

- **BGV.Enc**$_{pk,params}(m \in \mathcal{R}_2)$: The encryption works by running:

$$c = E.Enc_{A_L, params_L}(m).$$

  We note that the ciphertext could be augmented with an index indicating which level it belongs to.

- **BGV.Dec**$_{s_j, params_j}(c)$: Supposing that the ciphertext is under s'_j key, the decryption algorithm works by running:

$$E.Dec_{s_j}(c).$$

- **BGV.Add**$_{pk}(c_1, c_2)$: It takes two ciphertexts, which, without loss of generality, we can suppose that are encrypted under the same s_j (i.e. $c_i = BGV.Enc_{s_i, params_i}(m_i)$, for $i = 1, 2$). If they are not initially, one can use BGV.Refresh (defined below) to make it so.

  Compute $c_3 = c_1 + c_2 \ mod \ q_j$.

  Output $c_4 = BGV.Refresh(c_3, \tau_{s'_j \to s_{j-1}}, q_j, q_{j-1})$.

- **BGV.Mult**$_{pk}(c_1, c_2)$: It takes two ciphertexts to multiply them. We suppose that these ciphertexts are encrypted under the same s_j. If not, one can use use BGV.Refresh (defined below) to make it so.

  Compute $c_3 = c_1 \cdot c_2 \ mod \ q_j$.

  Output: $c_4 = BGV.Refresh(c_3, \tau_{s'_j \to s_{j-1}}, , q_j, q_{j-1})$.

- **BGV.Refresh**$(c, \tau_{s'_j \to s'_{j-1}}, q_j, q_{j-1})$: It takes two ciphertexts encrypted under s'_j, the auxiliary information $\tau_{s'_j \to s'_{j-1}}$ to facilitate key switching and the current and next modulus q_j and q_{j-1}. It proceeds as follows:

  1. Expand: Set $c_1 = PowersOf2(c, q_j)$.
  2. Switch Keys: Set $c_1 = SwitchKey(\tau_{s'_j \to s'_{j-1}}, c, q_j)$, a ciphertext under the key s_{j-1} for modulus q_j.
  3. Switch Modulus: Set $c_2 = Scale(c_1, q_j, q_{j-1}, 2)$, a ciphertext under the key s_{j-1} for modulus q_{j-1}.

Theorem 2 ([23], Theorem 3). For some $\mu = \theta(log\ \lambda + log\ L)$, BGV is a correct $L-$leveled scheme$-$ specifically, it correctly evaluates circuit of depth L with Add and Mult gates in $\mathcal{R}_2$. The per-gate computation cost is $\tilde{O}(d \cdot n_L^3 \cdot log_{q_j}^2) = \tilde{O}(d \cdot n_L^3 \cdot L^2))$. For the LWE case (where $d = 1$) the per-computation cost is $\tilde{O}(\lambda^3 \cdot L^5)$. For the RLWE case (where $n = 1$) the per-computation is $\tilde{O}(\lambda \cdot L^3)$.

To achieve a FHE from the BGV scheme presented above, one can combine the Refresh idea with the bootstrapping procedure.

3.3.3.3 BFV

The BV scheme was proposed by Fan and Vercauteren in 2012, based on the RLWE problem [24] by porting the scheme proposed by Brakerski [72] from the LWE instance to RLWE instance. Similar to [72], they make use of re-linearization, but their version is more efficient. Also, they use the modulus switching in order to simplify the bootstrapping method. The Gentry bootstrapping procedure without the squashing technique was used to turn this SHE scheme to FHE.

The *BFV* scheme = (ParamGen, KeyGen, Enc, Dec, Add, Mult) consists of the following algorithms:

- **BFV.ParamGen**$(\lambda) \to (n', q, t, \chi_{key}, \chi_{err}, w)$.
  It uses the parameter λ in order to fix a positive integer n', the modulus q and t (such that $1 < t < q$), the distributions χ_{key} and χ_{err} and finally an integer $w > 1$.

- **BFV.KeyGen**$(n', q, t, \chi_{key}, \chi_{err}, w) \to (pk, sk, rlk) = ((b, a), s, rlk)$. It works by:

  1. Sample: $s \xleftarrow{\chi} \mathcal{R}_q$, a $\xleftarrow{\$} \mathcal{R}_q$, and $e \xleftarrow{\chi} \mathcal{R}$.
  2. Compute $b = [-(a \cdot s + e)]_q$.
  3. Sample: $a' \xleftarrow{\$} \mathcal{R}_q^{l_{w,q}}$ and $e' \xleftarrow{\chi_{err}} \mathcal{R}_q^{l_{w,q}}$.

  For evaluation key:

– *Version 1*: It takes as (sk, T)[4]. For $i = 0, \dots \ell = \lfloor log_T(q) \rfloor$:

1. Sample: $a_i \xleftarrow{\$} \mathcal{R}_q$, $e_i \xleftarrow{\chi} \mathcal{R}_q$
2. Return

$$rlk = \left[\left(\left[-(a_i \cdot s + e_i) + T^i \cdot s^2 \right]_q, a_i \right) : i \in [0 \cdot \ell] \right].$$

– *Version 2*: It takes (sk, p) and performs the following steps:

1. Sample $a \xleftarrow{\$} \mathcal{R}_{p \cdot q}$, $e \xleftarrow{\chi'} \mathcal{R}_{p \cdot q}$
2. Return

$$rlk = \left(\left[-(a \cdot s + e) + p \cdot s^2 \right]_{p \cdot q}, a \right).$$

- **BFV.Enc**$_{pk}(m) \rightarrow c = (c_i, c_j, c_k = 0)$
The encryption algorithm takes the message $m \in \mathcal{M} := \mathcal{R}_t = \mathcal{R}/t\mathcal{R}$ and works as follow:

1. Sample: $u, e_1, e_2 \xleftarrow{\chi} \mathcal{R}$.
2. Return

$$c = ([p_0 \cdot u + e_1 + \Delta \cdot [m]_t]_q, [p_1 \cdot u + e_2]_q) \in \mathcal{R}_q^2.$$

- **BFV.Dec**$_{sk}(c) :\rightarrow m$. It takes the encrypted message and decryptes it by computing:

$$m = \left[\left\lfloor \frac{t}{q} \cdot [c_i + c_j s]_q \right\rceil \right]_t \in \mathcal{R}_t.$$

- **BFV.Add**$(c_1, c_2) :\rightarrow c_{add}$ s.t. $c_{add} = c_1 + c_2$
$c_{add} = \left([c_{1,i} + c_{2,i}]_q, [c_{1,j} + c_{2,j}]_q \right)$.
- **BFV.Mul**$_{rlk}(c_1, c_2) :\rightarrow c_{mul} = (c_i, c_j, c_k)$: It computes
$c_{mul} = \left(\left[\left\lfloor \frac{t}{q} \cdot c_{1,i} \cdot c_{2,i} \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q} \cdot (c_{1,i} \cdot c_{2,j} + c_{1,j} \cdot c_{2,j}) \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q} \cdot c_{1,j} \cdot c_{2,j} \right\rceil \right]_q \right)$
- **BFV.Relin**$_{rlk}(c_i, c_j, c_k) \rightarrow (c_i', c_j')$
  – *version 1:*
    1. Write c_k in base T, i.e. write $c_k = \sum_{i=0}^{\ell} c_k^{(i)} T^i$, with $c_k^{(i)} \in \mathcal{R}_T$
    2. Return

$$(c_i', c_j') = \left(\left[c_i + \sum_{n=0}^{\ell} rlk_{0,n} \cdot c_k^{(n)} \right]_q, \left[c_j + \sum_{n=0}^{\ell} rlk_{1,n} \cdot c_k^{(n)} \right]_q \right).$$

  – *version 2:*

[4] T is a chosen base (note that T is totally independent of t) where we can write c in base T i.e. $c = \sum_{i=0}^{\ell} T^i \cdot c_2^{(i)} mod q$ with $\ell = \lfloor log_T(q) \rfloor$ and $c_2^(i)$ are in $\mathcal{R}_T$.

1. Compute
$$(c_{k,0}, c_{k,1}) = \left(\left[\left\lfloor \frac{c_k \cdot rlk_0}{p} \right\rceil \right]_q, \left[\left\lfloor \frac{c_k \cdot rlk_1}{p} \right\rceil \right]_q \right)$$

2. Return
$$(c'_i, c'_j) = \left([c_i + c_{k,0}]_q, [c_i + c_{k,1}]_q \right).$$

Theorem 3. (from [24]) Using the notation of the scheme **BFV** and supposing that $\|\chi\| < B$, **BFV** can correctly evaluate a circuit of multiplicative depth **L** with:

$$4 \cdot \delta_R^L \cdot (\delta_R + 1.25)^{L+1} \cdot t^{L-1} < \lfloor q/b \rfloor. \tag{3.1}$$

where $\delta_R = max\{\|a \cdot b\|/(\|a\| \cdot \|b\| \ ) \ a, b \in R_q\}$ is the expansion factor.

In table 3.2 we present a comparison between the BFV and BGV cryptosystems in terms of key and ciphertext sizes. As explained in this table, we remark that the BGV scheme presents a big overhead in terms of the sizes of its keys and ciphertexts.

Scheme	Publick Key Size	Private Key Size	Ciphertext Size
BGV	$2 * d * n * log(q)$	$2 * d * log(q)$	$2 * d * log(q)$
BFV	$2 * d * log(q)$	d	$2 * d * log(q)$

Table 3.2: Public key, private key and ciphertext sizes for BGV and BFV scheme.

In this chapter, we present the first cryptographic tools used in our thesis to ensure data confidentiality. We describe the HE schemes used in our work. In the following chapter, we present the VC technique, more details on the VC schemes used in our thesis.

Chapter 4

Verifiable Computing

The need for outsourced computing or cloud computing increased significantly, after the rise of a new type of services, the "computing services", where users desire to reduce the charge of expensive computations by outsourcing any burdensome computational workload to a cloud or a service provider (SPs). The risks come with the growth of this type services. Therefore, we need to confirm that the result delivered by the cloud or SPs is correct which can be error-prone or otherwise not entirely trustworthy because the complex and large-scale of SPs structure. Consequently, an immediate need for result assurance naturally aroused. Therefore this resulted in the appearance of a new cryptography tool called *Verifiable Computing* (VC).

The main essential specialization of the VC is that the prover works to convince the verifier about the correctness of the delegated computation function for it.

4.1 VC approaches or Techniques

This section attempts to present a slight overview of the VC approaches. Our goal is not only to describe the related verifiable computing schemes but also to explain our position and the limits of the work we conducted.

A deep look in the literature on Verifiable computing problems concludes on the existence of two solution approaches, the first known as non-proof-based or hardware solutions and the second proof-based solutions.

4.1.1 Non-Proof-based and Hardware-based Solutions

The principle idea of this approach is that the verifier delegates some computation to multiple independent servers [91–93], and according to the results returned by these servers, the verifier decides the correctness of the result of this computation. For example, the SETI@Home project [91] uses the BOINC middleware [92] in order to validate the result. This middleware delegates the computation to different computers located in different nodes and compares the results output by these computers. If the results returned by these computers are matched, they are considered correct with high probability. Else, this middleware delegates the computation to the new computers until obtaining a matched result under the assumption that most computers are honest (applied the system correctly). Canetti et al. [93] improved this system to work with a single honest server among different servers.

Then this approach requires using several server providers (SP) where at least one is trusted since trusted hardware is usually strongly limited in scalability. In the following, we show the VC scheme using theoretical tools.

4.1.2 Proof-Based Solutions

First of all, we notice that this is not a detailed presentation of this approach. We present the general idea of each approach and several works related to it. We refer the interested reader to a survey conducted by Walfish and Blumberg [94], where they focus on general-purpose solutions that provide answers to the problem of VC for arbitrary functions.

The proposed VC schemes in this approach are composed of two entities: the verifier and the prover. The prover works to convinces efficiently the verifier about the correctness of a given computation which is represented as an NP-statement. To do this, in this approach the prover sends the result along with proof to determine that the result was computed correctly with a logical condition that the proof is inexpensive to check the result.

In the following, we present two types of proof-based VC solutions. In function of the level of confidentiality of the insert data, we distinguish two approaches according to the type of data of the delegated function: Proof-Based Solution over clear data, Proof-Based Solution over cipher data.

4.1.2.1 Proof-Based Solution over clear data

In this approach we differentiate between two principal approaches: *interactive proof-based solution* or *non-interactive proof-based solution.*

Indeed, we identify the two following characteristics used as follows:

4.1.2.1.1 Interactive Proof (IP) Based Solution. The Interactive Proofs system (IPs) was proposed for the first time by Babai [95] and Goldwasser et al. [42] where a verifier is a polynomial-time machine without any restrictions on prover. The verifier and the prover open a "dialogue" in order to ensure the goal of this approach, i.e. that the

prover convinces the verifier about the correctness of computation using a proof. In [96] Goldwasser et al. introduced the correct and sound proof concept (i.e. its not possible to return a proof for an incorrect computation) together with efficiency concept (the verifier's task is less than that of prover's). Nevertheless, such a system is very unpractical.

In order to put the IPs close to practicality of real scenario, Goldwasser et al. [96] transferred the concept of traditional IPs to the case of a quasi-linear verifier(super efficient) as well as the case of a super-polynomial time honest prover and a dishonest efficient prover. This IP is valid for any function representable as a log-space uniform boolean circuit that has communication complexity being the depth of the circuit. They proposed a preprocessing step to transform any function to such a circuit and the correctness of output for each gate is validated in an interaction with the verifier. Even if this approach is practical for the small circuit and functions that can be parallelized, other improvements were proposed in [97, 98]. We note that this proof system is publicly delegatable, which means that the entity who runs system setup can be different from the entities who form a task to be outsourced, but not publicly verifiable where publicly verifiable means that anybody can verify the correctness of the computation.

Probabilistically Checkable Proofs (PCP). In [99] Arora and Safra proposed a new VC solution, where a verifier is legitimately confident about the correctness of computing with "very high probability" and falsely convinced with "very small probability". In this setting, a proof (say of size n) is encoded, in such a way that a prover can convince the verifier about its correctness of computation (with high confidence level) using a quering to check a constant number of randomly chosen locations in this encoding (via interaction with prover). The PCP theorem [100, 101] stipulates that it's not necessary to query and verify the proof entirely (which can be very large and hard), to convince the verifier about the correctness of proof. This is also inefficient because the length of encryption can be so large and very hard. It yields more work both for the prover - to construct the proof and the verifier - to check at sufficient random locations in the encodings. Amelioration of this setting, particularly in the length of PCPs were proposed in [100, 101] but its still not deployable in "the real world", as well as soundness property is violated. The prover may be tempted to change the value of the query position in the certificate in order to answer the verifier's later query, while speciously matching the earlier query. Therefore, the proof calculated by the prover must be determined in advance in order to answer all the verifier's queries.

Argument Based Approach. Kilian introduced in 1992 [102] the idea of Interactive protocols that are sound against computationally bounded dishonest provers. In a nutshell, they combine the PCPs with linear commitments[1] and local openings. More in details, the idea is to commit to a PCP string π, i.e. the prover needs to send the commitment π to the verifier. Now the verifier can ask the prover to open the commitment in several positions. Then we obtain a four-move argument system. This idea is improved by Micali in [103] to turn this scheme into a one-move secure scheme. Several works based on this approach were proposed (like the Ishai et al. work [104]), which are more efficient in the sense of communication complexities and computation but not sufficient. The considered PCPs are linear functions, and the query verifier is a computation of function over

[1] A bit commitment protocol is a cryptographic protocol consisting of two parties, a sender and a receiver. The sender commits to the receiver to a bit b, such that the receiver does not know the value of b. Besides, the sender has no mean to change b after it was committed. Later on, the sender reveals bit b and the receiver can verify that b is really the committed bit.

some input selected by the verifier. Additionally, it uses a linear homomorphic encrypted scheme like Paillier [22] cryptosystem to issue a (linear) commitment to the proof.

Other directions for these approaches developed over the years such as: linear PCPs [104],Pepper [105], Ginger [106, 107], an improved work of Ishai et all. [104], etc.

4.1.2.1.2 Non-Interactive Solutions. These are new VC solutions without interaction between the prover and the verifier with the main characteristics being that the prover output the result of the delegated computation as well as proofs of correctness in the same message. The first non interactive proof based system was proposed by Micali in 2000 [103].

CS Proofs. Computationally Sound (CS) proofs as mentioned above were proposed by Micali [103]. They combine the rationale behind PCP with an efficient argument system, and uses the Fiat and Shamir heuristic [108] to eliminate the interaction between the prover and verifier. We note that the CS proofs are publicly delegatable and verifiable. Nevertheless, they rely on the random oracle model[2].

SNARKS. Bitanski et al. [109] defines the concept of non-interactive argument of knowledge (SNARK). The idea is to replace the random oracle used in the CS proof with an extractable collision-resistant hash function (ECRH). This setting relies on the unfalsifiable assumption[3] that for a given ECRH image, there is an extractor to calculate a pre-image. This scheme combines the theory behind CS proofs[103] with an instantiation of a Private Information Retrieval (PIR) protocol[4] (as the suggestion in [110]). This setting is publicly delegatable but relies on non-standard and non-falsifiable assumptions. As for the hardness Gentry and Wichs [109] display that it exists an intrinsic boundary on solutions based on SNARKs and they cannot rely on falsifiable assumptions[5].

Pinocchio Gennaro et al. [54] presented a way to construct succinct non-interactive arguments of knowledge (SNARKs) by means of QSPs and QAP. All these schemes are secure only under the non-falsifiable assumption. It also relies on a non-falsifiable assumption (i.e. knowledge of exponential power), which is a non-standard assumption.

However, several recent works proposed are proof-based approaches such as Geppetto: [111] (Generalized the QAP to MultiQAP, and how to reduce the server's overhead by decomposing circuits into a collection of subcircuits.), Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture [112] (new QAP based SNARK for arithmetic circuits that allows for more efficient verification and proof generation), etc.

VC From Attribute-Based Encryption (ABE) VC scheme was proposed by Parno et al. [113] to build a public delegation and public verification using an ABE scheme

[2]Random Oracle Model (ROM) is an oracle (a theoretical black box) that responds to every unique query with a (truly) random response chosen uniformly fromits output domain. If a query is repeated, it responds the same way every time that query is submitted.

[3]Defined by Gentry and Wichs [51], its proposed as a game model between a challenger and an adversary under falsifiable assumptions.

[4]PIR protocol is a protocol that allows a user to retrieve an item from a server in possession of a database without revealing which item is retrieved.

[5]A falsifiable assumption can be modeled as an interactive game between an efficient challenger and an adversary at the conclusion of which the challenger can efficiently decide whether the adversary "won" the game.

[114][6]. Their construction verifies that a function $f : \{0,1\}^n \rightarrow \{0,1\}$ can be computed by a polynomial sized Boolean formula. They use the attribute idea, meaning that the decryption of the encrypted message is successful under an attribute x iff $f(x) = 1$ holds. We note that this setting does not provide input/output privacy, and the security has not been analyzed yet.

4.1.2.2 Proof-Based Solution over encrypted data

VC on/from homomorphic encrypted data. A specific type of VC solutions, where the prover works to convince the verifier on the correctness of function evaluated about homomorphic encrypted data. These VC protocols use the FHE schemes as a building block. We note that these schemes are privately verifiable and ensure both input and output privacy.

Gennaro et al. [56] formalize the concept of non-interactive verifiable computation by combining the garbled circuits[115, 116] with FHE [19], in the amortized model[7] that allows reusing the garbled circuit multiple times for multiple verification, nonetheless preserving security. The idea of this solution is to represent function f (the function to be outsourced) as a garbled circuit C that associates random labels to each wire in the circuit, by the verifier that generates also the labels associated with an input. The prover computes the label associated to calculation with the output of function as a response. Now, the verifier uses the input, the input labels, the output of the garbled circuit to verify the correctness of the computation delegated. In addition to ensuring confidentiality for this construction, Gennaro et al. were the first to present a formal definition for the notion of verifiable computation that will be presented in section 4.2.

Another solution of this kind was the one by Chung et al. [117], which proposed another way to verify the correctness of a result using FHE in the amortized efficiency model while the server's overhead depends on the underlying FHE scheme, but in terms of security, it ensures weaker security. In [118] a similar scheme with a reduction in the preprocessing phase was presented but it's a weak security scheme.

A special work in this category is the work of Fiore et al. [3], where they used the MAC[8] to construct a verifiable computing scheme of multivariate polynomial with degree at most 2. This scheme is one of our principal pillars of our thesis, as described in detail in the subsection 4.4.1. We can also classify this work under the class of verifiable computing built using the MAC tool.

VC on/from Homomorphic Message Authentication Codes (MAC). Another line of research to find a practical VC is using the cryptographic tools Message Authentication Codes [119, 120] that allows for private verification where the owner of a secret key can verify the authenticators. The Homomorphic MAC was defined for the first time by Gennaro and Wichs in [119]. With this structure, the prover can ensure the correctness of arbitrary functions (particularly Boolean functions) on authenticated data, where the prover can produce (homomorphically) an unforgettable tag that validates the correctness

[6] ABE is a public-key encryption in which the secret key of a user and the ciphertext are dependent upon attributes .

[7] The verifier in the amortized model must perform one-time expensive preprocessing operations to allow an unlimited number of valid verifications.

[8] Message Authentication Code (MAC) which is a cryptography tool to ensure that the received message has not been changed in the way, then it ensures the integrity of data.

of the computation, without deploying any secret key. Its used only to check the validity of tag that certifies the correctness of the computation. This setting used also FHE in order to ensure data confidentiality.

There are several ways to ensure VC schemes using cryptographic tools like Homomorphic signature[9](whereby anyone have access to the verifying public key can verify a homomorphic signature without knowledge of the secret signing key) [50], AD-SNARK[121] (combine QAP idea with linear homomorphic MACs).

VC for specified function. Other VC schemes proposed to address a specific class of functions such as matrix computation [122–124], polynomial evaluation in [122, 123, 125, 126], and keyword search [125, 127].

Recently, in the middle of my thesis, new VC schemes were published. More precisely in 2020, Fiore et al. [58] extended the approach of [3] to evaluate more than quadratic functions (yet, of fixed degree) by means of zkSNARKs of polynomial rings as well as the HE scheme (an instance, over BGV scheme [23]) with particular a prime modulus q (bigger than 2^{λ}) used in the zkSNARK. This is an efficient proof generation, but it lacks an implementation, and it requires a relatively large value for the ciphertext modulus.

Another scheme proposed in 2021 by Bois et al. [128] ensures the correctness of functions over encrypted data while simplifying the specification of the q parameter in the Fiore et al. work [58], i.e. it allows a flexible choice of HE parameters.

Still, these two schemes have been limited, in the sense of HE improvement. For example, they don't support speedups through classical optimizations such as the, Residue Number System[10] (RNS).

4.2 Background

In this section, we present a formal definition of a verifiable computing scheme and their relevant properties. We will define the VC in the following section according to the model in [56] and taking into account the updates from Fiore et al. [3].

4.2.1 Problem Definition

Verifiable computing techniques allow to prove and verify the integrity of computations on authenticated data. A Verifiable Computation scheme is defined as a protocol in which a client (usually weak) has a function f and some data denoted x and delegates to another client (in most cases a server) the computation of $y = f(x)$. Then the same client or another one can receive the result y plus a short proof of its correctness. More in details, a user generates an authentication tag σ_x associated with his/her data x with his/her secret key and the server computes an authentication tag $\sigma_{f,y}$ that certifies the value $y = f(x)$ as an output of the function f. Now, anyone using the verification key (public or secret) can verify y to check that y is indeed the result of $f(x)$.

[9]The Homomorphic signatures is the "public" version of the homomorphic MAC scheme.

[10]Residue Number System: is a numeral system for representing an integer by their values modulo several pairwise coprime integers, a method used for improving the speedup of the calculation for HE scheme.

A verifiable computation scheme VC = (**KeyGen,ProbGen,Compute,Verify**) consists of the four following algorithms:

- (PK,SK)←**KeyGen**(f, λ): Taking as input the security parameter λ and a function f, this randomized key generation algorithm generates a public key (that encodes the target function f) used by the server to compute f. It also computes a matching secret key, kept private by the client.
- (σ_x,τ_x)←**ProbGen**$_{SK}(x)$: The problem generation algorithm uses the secret key SK to encode the input x as a public value σ_x, given to the server to compute with, and a secret value τ_x which is kept private by the client.
- σ_y ←**Compute**$_{PK}(\sigma_x)$: Using the client's public key and the encoded input, the server computes an encoded version of the function output $y = f(x)$
- (acc, y) ←**Verify**$_{SK}(\tau_x,\sigma_y)$: Using the secret key SK and the secret τ_x, this algorithm converts the server output into a bit acc and a string y. If $acc = 1$ we say that the client accepts $y = f(x)$, meaning that the proof is correct, else (i.e. $acc = 0$) we say the client rejects it.

4.2.2 Properties of VC

In this section, we recall the main properties for a verifiable computation scheme, as defined in [56] and [3]: correctness, privacy, outsourceability with verification queries by the adversary, function privacy (capability of the scheme to hide f from the server) and adaptive security.

Definition 8 (**Correctness**). A VC scheme is correct if, for all (f, x), with:

- $(PK, SK) \leftarrow$ **Key Gen**(f, λ),
- (σ_x,τ_x)←**ProbGen**$_{SK}(x)$,
- σ_y ←**Compute**$_{PK}(\sigma_x)$.

Then

$$\textbf{Verify}_{SK}(\tau_x, \sigma_y) \rightarrow (1, y = f(x)).$$

When verifiable computing is correct for a given function f and input x, a malicious server has a negligible probability to convince the one running the verification algorithm to accept a wrong output ŷ, i.e. **Verify**$_{SK}(\tau_x, \sigma_{\hat{y}}) \rightarrow acc = 0$ for ŷ$\neq f(x)$.

In order to introduce the other notions, we first need to define the following oracles.

- **PProbGen**$_{SK}(x)$: runs **ProbGen**$_{SK}(x)$ to get (σ_x,τ_x) and returns only σ_x.
- **PVerify**$_{SK}(\tau, \sigma)$: returns acc of **Verify**$_{SK}(\tau,\sigma)$ (public acceptance/rejection bit resulting from a verification request).

The following experiment is used to explain the notion of security for a VC scheme. Note that, in this experiment, $poly(.)$ is a polynomial on its inputs and A is an adversary allowed to query $PVerify_{SK}(\tau, .)$ with τ a secret encoding obtained with $PProbGen_{SK}$.

Experiment $\textbf{Exp}_A^{Verif}[VC, f, \lambda]$;

(PK,SK)$\leftarrow$**KeyGen**(f, λ);

For $i = 1, \ldots, \ell = poly(\lambda)$:

$x_i \leftarrow A(PK, x_1, \ldots, x_{i-1}, \sigma_{i-1})$;

$(\sigma_i,\tau_i)\leftarrow$**ProbGen**$_{SK}(x_i)$;

$(i, \hat{\sigma}_y) \leftarrow A(PK, x_1, \sigma_1, \ldots, x_\ell, \sigma_\ell)$;

$(\hat{acc}, \hat{y}) \leftarrow$**Verify**$_{SK}(\tau_i,\hat{\sigma}_y)$;

If $\hat{acc} = 1$ and $\hat{y} \neq f(x_i)$, output '1' else output '0'.

In this experiment, A has access to the above two oracles to generate the encoding of many problem instances and, to check, arbitrarily, the response of the client. If A is able to convince the verifier of the output she produced even if its incorrect, then A is successful.

A verifiable computing scheme VC is correct for a function f and an adversary A running in probabilistic polynomial time(PPT), if

$$Prob[\mathbf{Exp}_A^{Verif}[VC, f, \lambda] = 1] \leq neg(\lambda) \tag{4.1}$$

where $neg()$ is a negligible function on its input. This probability is the advantage of A denoted Adv_A^{Verif}.

For efficient Verifiable Computing schemes, the time to encode an input and verify an output have to be smaller than the time to compute the function from scratch. This corresponds to the **outsourceability property**, defined as follows:

Definition 9. (**Outsourceability**) A VC can be outsourced if it allows efficient generation and verification. So, for any x and σ_y, the time required for **ProbGen**$_{SK}(x)$ plus the time required for **Verify**$_{SK}(\sigma_y)$ is $o(T)$, where T is the time required to compute $f(x)$.

The definition of input privacy, based on a typical indistinguishability argument, will guarantee that no information about the inputs are leaked. If VC is private for input, output privacy follows naturally.

Intuitively, a verifiable computation scheme is private when the public outputs of the problem generation algorithm **ProbGen** over two different inputs are indistinguishable. For a VC with function privacy, the public key generated with **Keygen** should not reveal any information on the encoding of f even for an adversary with polynomial runs of **ProbGen**$_{SK}$ on chosen inputs.

More formally, to define the privacy and the function privacy, we consider the following experiments.

Experiment $\mathbf{Exp}_A^{Priv}[VC, f, \lambda]$

$(PK, SK) \leftarrow$**KeyGen**(f,λ);

$b \leftarrow \{0, 1\}$;

$(x_0, x_1) \leftarrow \mathbf{A}^{\mathbf{PVerify,PProbGen}}(PK)$;

$(\sigma_0,\tau_0)\leftarrow$**ProbGen**$_{SK}(x_0)$;

$(\sigma_1, \tau_1) \leftarrow$**ProbGen**$_{SK}(x_1)$;

$\hat{b} :\leftarrow A^{PVerify,PProbGen}(PK, x_0, x_1, \sigma_b)$;

if $\hat{b} = b$ then output '1', else '0'.

Experiment $\mathbf{Exp}_A^{FPriv}[VC, \lambda]$

$(f_0, f_1) \leftarrow \mathcal{A}(\lambda)$;

$b \leftarrow \{0, 1\}$;

$(PK, SK) \leftarrow \mathbf{KeyGen}(f_b, \lambda)$;

For $i = 1, \ldots, l = poly(\lambda)$:

$x_i \leftarrow$

$\mathbf{A}^{\mathbf{PVerify}}(PK, x_1, \sigma_1, \ldots, x_{i-1}, \sigma_{i-1})$;

$(\sigma_i, \tau_i) \leftarrow \mathbf{ProbGen}_{SK}(x_i)$;

$\hat{b} :\leftarrow \mathbf{A}^{\mathbf{PVerify}}(PK, x_1, \sigma_1, \ldots, x_\ell, \sigma_\ell, \sigma_b)$;

if $\hat{b} = b$ then output '1', else '0'.

Definition 10. (**Privacy**) A verifiable computing VC is private for a function f if, for any probabilistic polynomial time (PPT) adversary A,

$$Prob[\mathbf{Exp}_A^{Priv}[VC, f, \lambda] = 1] \leq \frac{1}{2} + neg(\lambda) \tag{4.2}$$

Definition 11. (**Function Privacy**) A verifiable computing VC is function private if, for any PPT adversary A,

$$Prob[\mathbf{Exp}_A^{FPriv}[VC, \lambda] = 1] \leq \frac{1}{2} + neg(\lambda) \tag{4.3}$$

In [3], adaptive security for a VC scheme is defined as the security if the adversary chooses f after having seen many "encodings" of σ_x for adaptively-chosen values x. As such, one has to first specify the type of schemes allowing to compute σ_x before choosing f.

Definition 12. (**Split Scheme**) A scheme of verifiable computing is a split scheme if the following conditions hold:

- There exists PPT algorithms $\mathbf{KeyGen}^E(\lambda)$, $\mathbf{KeyGen}^V(f, \lambda)$ such that: if $(PK, SK) \leftarrow \mathbf{KeyGen}(f, \lambda)$, then $PK = (PK_E, PK_V)$ and $SK = (SK_E, SK_V)$, where: $\mathbf{KeyGen}^E(\lambda) \rightarrow (PK_E, SK_E)$ and $\mathbf{KeyGen}^V(f, \lambda, PK_E, SK_E) \rightarrow (PK_V, SK_V)$.
- There exist PPT algorithms $\mathbf{ProbGen}^E_{SK_E}(x)$, $\mathbf{ProbGen}^V_{SK_V}(x)$ such that: if $(\sigma_x, \tau_x) \leftarrow ProbGen_{SK_E, SK_V}(x)$, then $\sigma_x = \sigma_x^E \leftarrow \mathbf{ProbGen}^E_{SK_E}(x)$ and $\tau_x = \tau^V \leftarrow \mathbf{ProbGen}^V_{SK_V}(x)$.

For a split scheme, and for any delegated f one can generate a valid σ_x before knowing f (because σ_x is independent of f). For this, one can run $\mathbf{KeyGen}^E(\lambda)$ to obtain (PK_E, SK_E), and setting $\sigma_x \leftarrow \mathbf{ProbGen}^E_{SK_E}(x)$ before knowing f. The validity of σ_x is true for all $(PK, SK) = ((PK_E, PK_V), (SK_E, SK_V))$.

Let us now describe the following experiment necessary to define the adaptive security for split schemes.

Experiment $\mathbf{Exp}_A^{Adap-Verify}[VC, \lambda]$

$(PK, SK) \leftarrow \mathbf{KeyGen}^E(\lambda)$;

For $i = 1, \ldots, \ell' = poly(\lambda')$:

$x'_i \leftarrow \mathbf{A}(PK_E, x'_1, \sigma'_1, \ldots, x'_{i-1}, \sigma'_{i-1})$;

$\sigma'_i \leftarrow \mathbf{ProbGen}^E_{SK_E}(x_i)$;

$f \leftarrow A(x'_1, \sigma'_1, \ldots, x'_{\ell'}, \sigma'_{\ell'})$;

$(PK_V, SK_V) \leftarrow \mathbf{KeyGen}^V(f, \lambda)$;

$(PK, SK) \leftarrow (PK_E, PK_V, SK_E, SK_V)$;

For $i = 1, \ldots, \ell = poly(\lambda)$:

$x_i \leftarrow \mathbf{A}^{\mathbf{PVerify}}(PK_E, x_1, \sigma_1, \ldots, x_{i-1}, \sigma_{i-1})$;

$(\sigma_i, \tau_i) \leftarrow \mathbf{ProbGen}_{SK}(x_i)$; $(\hat{acc}, \hat{y}) \leftarrow \mathbf{Verify}_{SK}(\tau_i, \hat{\sigma}_i)$;

$(i, \hat{\sigma}_y) \leftarrow \mathbf{A}^{\mathbf{Pverify}}(PK, x_1, \sigma_1, \ldots, x_\ell, \sigma_\ell)$; if $\hat{acc} = 1$ and $\hat{y} \neq f(x_i)$ output '1', else '0';

Definition 13. (**Adaptive security**) A verifiable computing VC is adaptively secure, if, for any PPT adversary A,

$$Prob[\mathbf{Exp}_A^{adap-verif}[VC, \lambda] = 1] \leq \frac{1}{2} + neg(\lambda) \tag{4.4}$$

4.3 Preliminary tools

Before presenting a detailed description of verifiable computing used in our work, we present two required tools in the construction of Fiore et al. VC [3]: *Pseudo Random Function with Amortized closed-form Efficient*, and *Homomorphic Hash function.*

4.3.1 Homomorphic Hash Function

Informally, a family of homomorphic hash functions **H** with domain $\mathcal{X}$ and range $\mathcal{R}$ consists of three algorithms (**H.KeyGen, H, H.Eval**). The first one, the key generation hash **H.KeyGen**, generates the description of the hash function H_K,where K is the key, the function **H** computes the hash and, finally, **H.Eval** allows the computation over $\mathcal{R}$ satisfying the following homomorphic property:

$\mathbf{H.Eval}(f, (\mathbf{H}(x_1), \ldots, \mathbf{H}(x_n))) = \mathbf{H}(f(x_1, \ldots, x_n)),\ x_i \in \mathcal{X}$(**H** is a ring homomorphism).

Realizations of A Collision-Resistant Homomorphic Hash [3]

We present a realization of homomorphic hash $\hat{\mathbf{H}}$ that permits to reduce a BGV encrypted $\mu \in R_q[y]$ ($\mu_i = \text{BGV.Enc}_{PK}(m_i)$) into a $v \in \mathbb{Z}/q\mathbb{Z}$ depending on the degree of μ denoted $deg_y(\mu)$ while preserving the homomorphic property ($\hat{H}.Eval(f, (\hat{H}(\mu_1), \ldots, \hat{H}(\mu_n))) = \hat{H}(f(m_1, \ldots, m_n))$ with m_i a BGV plaintext). We point out that this realization is done by Fiore et al. [3]

In their construction of $\hat{\mathbf{H}}$, they used the parameters $bgpp = (q, g, h, e)$ where q be a prime of λ bits, as well as a homomorphic hash function $(H.KeyGen, H, H.Eval)$ with domain $R_q[y]$ and range $\mathbb{F}_q = \mathbb{Z}/q\mathbb{Z}$, which is described as follows:

$$\begin{cases} \mathbf{H.KeyGen}: \text{ select } (\alpha, \beta) \in R_q \times \mathbb{Z}/q\mathbb{Z} \text{ and set } \kappa = (\alpha, \beta). \\ \mathbf{H}_\kappa(\mu \in R_q[y]) : \text{evaluates } \mu \text{ at } Y = \alpha \text{ and evaluate } \mu(\alpha) \text{ at } \beta. \\ \mathbf{H.Eval}(f_g, \nu_1, \nu_2): \text{ compute } f_g(\nu_1, \nu_2) \text{ where } f_g \text{ is } + \text{ or } \times. \end{cases}$$

For $\mu \in \mathcal{D} = \{\mu \in \mathbb{Z}_q[x][y] : deg_x(\mu) = N, deg_y(\mu) = c\} \subset R_q[y]$, $H_{\alpha,\beta}(\mu)$ consists of evaluating μ at $y = \alpha$, and then evaluating $\mu(\alpha)$ at β i.e. $H_{\alpha,\beta}(\mu) = ev_\beta \circ ev_\alpha(\mu)$.

The authors of [3] demonstrated that the above **H** is homomorphic and universal one-way. More precisely, for all $\mu, \mu' \in \mathcal{D}$, such that $\mu \neq \mu'$:

$$Pr[H_\kappa(\mu) = H_\kappa(\mu') : (\alpha, \beta) \in R_q \times \mathbb{Z}/q\mathbb{Z}] \leq \frac{c + N}{q}.$$

which is negligible for an appropriate choice of $q \approx 2^{\lambda}$. But its secure only if the key$\kappa = (\alpha, \beta)$ is kept secret and the function is used only one time(otherwise information on α and β is leaked). For this reason, the authors proposed a new version of **H** marked as $\hat{H}$, which is a proven collision-resistant homomorphic hash preserving the homomorphic property but only for the functions of degree 2.

- **$\hat{\mathbf{H}}$.KeyGen**$\rightarrow (K, \kappa = (\alpha, \beta))$:
  First at all, generate $bgpp = (g, h, q)$,
  Next, sample a random $(\alpha, \beta) \leftarrow (\mathbb{F}_q)^2$. Afterwords, for $i = 0, \ldots, c$, $j = 1, \ldots, N$, we calculate $g^{\alpha^i \beta^j}$, and $h^{\alpha_i \alpha_j}$ and include them to K.
  Output K and $\kappa = (\alpha, \beta)$.
- **$\hat{\mathbf{H}}$**: For $\mu \in \mathcal{D}$, in function of its degree $deg_y(\mu)$, $\hat{H}_\kappa(\mu)$ is computed differently.

  If $deg_y(\mu) \leq 1$ then $\hat{H}_\kappa(\mu) = (T, U) = (g^{H_\kappa(\mu)}, h^{H_\kappa(\mu)}) \in \mathbb{G}_1 \times \mathbb{G}_2$.

  If $deg_y(\mu) = 2$, then $e(g, h)^{H_\kappa(\mu)}$.
- **$\hat{\mathbf{H}}$.Eval**(f_g, ν_1, ν_2): It computes in a homomorphic way a function of degree 2 on the outputs of $\hat{H}$.

  For $\nu_1 = (T_1, U_1)$, $\nu_2 = (T_2, U_2)$ and (respectively, $\hat{\mathrm{T}}_1$, $\hat{\mathrm{T}}_2 \in \mathbb{G}_T$).

$$\begin{cases} \nu_1 + \nu_2 = (T_1 \cdot T_2, U_1 \cdot U_2) \text{ (resp } \hat{T} \leftarrow \hat{T_1} \cdot \hat{T_2}). \\ c \cdot \nu = (T^c, U^c) \text{ (resp } \hat{T^c}) \text{ for } c \in \mathbb{F}_q. \\ \nu_1 \cdot \nu_2 = e(T_1, U_2) \in \mathbb{G}_T. \end{cases}$$

Fiore et al [3] demonstrated that this hash $\hat{H}$ is homomorphic and its collision-resistant under the $\ell-$BDHI assumption[11].

Theorem 4. The function $\hat{H}$ described above is homomorphic. Furthermore, if the $\ell-$BDHI assumption holds for $\mathcal{G}$ (for any $\ell \geq N, c$), then $\hat{H}$ is collision-resistant. More precisely, for $(K, \kappa) \leftarrow \hat{H}$.**KeyGen** :

$$Pr[\hat{H}(\mu) = \hat{H}(\mu') \wedge \mu \neq \mu' | (\mu, \mu') \rightarrow \mathcal{A}(K)] = neg(\lambda).$$

We refer the one interested in this proof for this theorem to the theorem 3 in [3] .

4.3.2 Pseudo Random Function with Amortized closed-form Efficient

Now, let us present the notion of *Pseudo Random Function(PRF)* with Amortized Closed-Form Efficiency [55], as well as its security notion.

Definition 14. [55] Consider a computation Comp that takes as input n random values $R_1, \ldots, R_n \in \mathcal{R}$, and a vector of m arbitrary values $z = (z_1, \ldots, z_m)$, and assume that the

[11]The $\ell-$Bilinear Diffie−Hellman Inversion ($\ell-$BDHI) Problem is defined as follows: Let $\mathcal{G}$ be a bilinear map generator, and let $bgpp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \mathcal{G}(\lambda)$. Let z be chosen uniformly at random in $\mathbb{Z}_q$. We say that the $\ell - \ell - BDHI$ assumption holds for $\mathcal{G}$ if for every PPT adversary $\mathcal{A}$ and any $\ell = poly(\lambda)$ the probability $P\left[\mathcal{A}(bgpp, g, h, g^z, h^z \ldots, g^{z^\ell}, h^{z^\ell}) = e(g, h)^{1/z}\right] = neg(\lambda)$.

computation of $Comp(R_1, \ldots, R_n, z_1, \ldots, z_m)$ requires time $t(n, m)$. Let $L = (L_1, \ldots, L_n)$ be arbitrary values in the domain χ of $\mathbf{F}$ such that each one can be interpreted as $L_i = (\Delta, \tau_i)$.

We say that a **PRF (KG, F)** satisfies amortized closed-form efficiency for $(Comp, L)$ if there exist two algorithms $\mathbf{CFEval}^{off}_{Comp,\tau}$ and $\mathbf{CFEval}^{on}_{Comp,\Delta}$ such that:

1. Given $w \leftarrow \mathbf{CFEval}^{off}_{Comp,\tau}(K, z)$ we have that:

$$\mathbf{CFEval}^{on}_{Comp,\Delta}(K, w) = Comp(\mathbf{F}_K(\Delta, \tau_1), \ldots, \mathbf{F}_K(\Delta, \tau_p), z_1, \ldots, z_m).$$

2. the running time of $\mathbf{CFEval}^{on}_{Comp,\Delta}(K, w)$ is $O(t)$[12].

Definition 15. A **PRF (F.KG, F)** is secure if, for every PPT adversary $\mathcal{A}$, we have that: $\left|Pr[\mathcal{A}^{F_K(\cdot)}(\lambda, pp) = 1] - Pr[\mathcal{A}^{\Phi(\cdot)}(\lambda, pp) = 1]\right| \leq neg(\lambda)$ where: $(K, pp) \leftarrow KG(\lambda)$ and $\Phi : \chi \rightarrow \mathcal{R}$ is a random function (i.e.its not possible to distinguish between F and Φ).

Realization of PRF with amortized closed-form efficiency.

Now we present the realization for the PRF with amortized closed-form efficiency, done by Fiore et al. [3], an adaptation of the scheme of Bakes et al in [55] to work with the asymmetric bilinear groups. The authors proposed this realization to acquire efficiency for its VC schemes for quadratic multi-variate polynomials.

Let $f : \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ be an arithmetic circuit of degree 2, and, without loss of generality, parse $f(x_1, \ldots, x_n) = \sum_{i,j}^{n} \zeta_{i,j} \cdot x_i \cdot x_j + \sum_{k=1}^{n} \zeta_k \cdot x_k$.

For some $\zeta_{i,j}, \zeta_k \in \mathbb{F}_q$, it defines $\hat{f} : (\mathbb{G}_1 \times \mathbb{G}_2)^n \rightarrow \mathbb{G}_T$ as the compilation of f on group elements such as: $\hat{f}(A_1, B_1 \ldots, A_n, B_n) = \prod_{i,j}^{n} \zeta_{i,j} \cdot e(A_i, B_j) \cdot \sum_{k=1}^{n} \zeta_k \cdot e(A_k, h)$.

- $\mathbf{F.KG}(\lambda) \rightarrow K = (K_1, K_2)$:
  First generate $bgpp = (q, g, h, e)$ some bilinear group parameters, where $\mathbb{G}_1 =< g >, \mathbb{G}_2 =< h >$ *and* $q = order(\mathbb{G}_i)$ *for* $i = 1, 2$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ a non−degenerate($\mathbb{G}_T =< e(g, h) >$) bilinear map. Choose two seeds K_1, K_2 for a family of PRFs $\mathbf{F}'_{K_{1,2}} : \{0,1\}^* \rightarrow \mathbb{F}_q^2$. Output K_1, K_2. The parameters define $F : \chi = \{0,1\}^* \times \{0,1\}^* \rightarrow \mathcal{R}^3$.
- $\mathbf{F}_K(\Delta, \tau) \rightarrow (R, S, V)$: It generates $(u, v) \leftarrow F'_{K_1}(\tau)$ and $(a, b) \leftarrow F'_{K_2}(\Delta)$ Finally it calculates $(R, S) = (g^{ua+vb}, h^{ua+vb})$.
- $\mathbf{CFEval}^{off}_{\tau}(K, f) \rightarrow w_f = \rho$: For $i = 1$ *to* t : calculate $(u_i, v_i) = F'_{K_1}(\tau_i)$ and construct a linear map ρ_i using (u_i, v_i) as $\rho_i(x_1, x_2) = u_i \cdot x_1 + v_i \cdot x_2$ Run $\rho \leftarrow f(\rho_1, \ldots, \rho_t)$, i.e., $\forall z_1, z_2 \in \mathbb{F}_q$: $\rho(z_1, z_2) = f(\rho_1(z_1, z_2), \ldots, \rho_t(z_1, z_2))$.
- $\mathbf{CFEval}^{on}_{\Delta}(K, w_f) \rightarrow W$: It generates $(a, b) \leftarrow F'_{k_2}(\Delta)$ and computes $W = e(g, h)^{w_f(a,b)}$.

This realization $\mathbf{F}$ is a pseudorandom function with amortized closed-form efficiency for $Comp = \hat{f}$, and its secure under the decision linear [13] [26] assumption in asymmetric bilinear groups. These properties are proved in [3], especially, in the theorem 4.

[12] Big O Notation is the language we use to describe the complexity of an algorithm. We express the runtime in terms of how quickly it grows relative to the input, as the input gets larger.

[13] The decision linear assumption holds for $\mathcal{G}$ if for every PPT algorithm A, $Adv^{dlin}_{\mathcal{A}}$ is negligible. Where The $Adv^{dlin}_{\mathcal{A}}$ defined as: Let $bgpp \xleftarrow{\$} \mathcal{G}$. Let $r_0, r_1, r_2, x_1, x_2 \leftarrow \mathbb{Z}_q$ be chosen uniformly at random. Let $T =$

4.4 VC schemes

4.4.1 VC for Quadratic polynomials on BGV Encrypted data

In this section, we present the VC scheme of Fiore et al. [3] for the case of multi-variate polynomials of degree 2, over BGV encrypted data. First the client encrypts his/her data $x = (x_1, \ldots, x_n)$ as a BGV ciphertext, where the plaintext modulus q is chosen to be prime. In parallel with the encryption of x_i, he/she also generates a tag σ_i for his/her data, using the combination of the PRF output and the hash collision-resistant functions (that compresses a BGV ciphertext into a double group elements). Once the server receives the BGV ciphertexts $[x_i]_{BGV}$ and the tags σ_i from the user, it computes f over $[x_i]_{BGV}$ and over σ_i and it obtains $y = f([x_i]_{BGV})$ and respectively a tag $\sigma = f(\sigma_i)$. The server sends y with the associated tag to the user owning the verification keys, which then checks the output in constant time (because he has already done a pre-computation phase).

In this scheme, we require to authenticate:

1. Each of the $2n' - \mathbb{F}_q$ components of a BGV ciphertext;
2. The BGV evaluation circuit $\hat{f} : \mathbb{F}_q^{2nn'} \to \mathbb{F}_q^{3n'}$ instead of $f : \mathbb{F}_q^n \to \mathbb{F}_q$.

More formally, our VC scheme is specified by the following algorithms:

1. **KeyGen**$(f, \lambda) \to (PK, SK)$, with the following steps:
   - Generate $bgpp = (q, g, h, e)$ some bilinear group parameters, where $\mathbb{G}_1 =< g >$, $\mathbb{G}_2 =< h >$, $q = order(\mathbb{G}_1) = order(\mathbb{G}_2)$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a non-degenerate bilinear map ($\mathbb{G}_T =< e(g, h) >$).
   - Run BGV.Setup$(\lambda) \to (n', q, t, \chi_{err}, \chi_{key}, w)$ to generate the parameters for the BGV encryption scheme. Run BGV.KeyGen()$\to (pk, sk, evk)$.
     Recall that the ciphertext of BGV scheme is $R_q := \mathbb{F}_q[X]/\Phi_m(X)$ be the polynomial ring where $\Phi_m(X)$ is the mth cyclotomic polynomial in $\mathbb{F}_q[X]$ of degree $n' = \Phi(m)$, the cleartext $\mathcal{M}$ is the ring $R_q[Y]$.
   - Run $\hat{H}.KeyGen \to (\kappa, \hat{K})$ to choose a random member of the hash function family $\hat{H} : \mathcal{D} = \{\mu \in \mathbb{Z}_q[x][y] : deg_x(\mu) \leq 2(n' - 1)\ , deg_y(\mu) \leq 2\} \subset R_q[y] \to \mathbb{G}_1 \times \mathbb{G}_2$. In this scheme, we do not use the public key of $\hat{H}$, so its not necessary to calculate it.
   - Sample a random value $r \leftarrow \mathbb{F}_q$.
   - Run PRF.KeyGen$(\lambda) \to (K, pp)$ to build $F_K : \{0,1\}^* \to \mathbb{G}_1 \times \mathbb{G}_2$. In this scheme, we need F_K to be computationally indistinguishable from a function that outputs $(R, S) \in \mathbb{G}_1 \times \mathbb{G}_2$ such that $Dlog_g(R) = Dlog_h(S)$ [14] is uniform over $\mathbb{F}_q$ (i.e. $e(R, h) = e(g, S)$).
   - Run $CFEval_\tau^{off}(K, f) \to w_f$, called the concise information of f.

$(g, h, g^{x_1}, g^{x_2}, g^{x_1 r_1}, g^{x_2 r_2}, h^{x_1}, h^{x_2}, h^{x_1 r_1}, h^{x_2 r_2})$. We define the advantage of an adversary $\mathcal{A}$ in solving the decision linear problem as $Adv_{\mathcal{A}}^{dlin} = |Pr[\mathcal{A}(bgpp, T, g^{r_1+r_2}, h^{r_1+r_2}) = 1] - Pr[\mathcal{A}(bgpp, T, g^{r_0}, h^{r_0}) = 1]|$

[14] Dlog is the Discrete logarithms problem where The discrete logarithm to the base g of R in the group G_1 is defined to be x where $R = g^x$ in G_1.

- Set $SK = (pk, sk, \kappa, r, K, w_f)$ and $PK = (pk, pp, f)$.

2. **ProbGen**$_{SK}(\overrightarrow{x} = (x_1, \ldots, x_n)) \rightarrow \sigma_x, \tau_x$, requiring the operations below:
   - Choose an arbitrary string $\Delta \in \{0,1\}^\lambda$ (identifier for $\overrightarrow{x}$).
   - For i=1 to n:
     (a) Run BGV.Enc(x_i)=$\mu_i \in R_q^2$ and compute its hash value $(T_i, U_i) = \hat{H}_\kappa(\mu) \in \mathbb{G}_1 \times \mathbb{G}_2$. Next run $\mathrm{F}_K(\Delta, i) = (R_i, S_i) \in \mathbb{G}_1 \times \mathbb{G}_2$.
     (b) Compute $X_i = (R_i \cdot T_i^{-1})^{1/r}$ and $Y_i = (S_i \cdot U_i^{-1})^{1/r} \in \mathbb{G}_1, \mathbb{G}_2$ respectively.
     (c) Set $\sigma_i = (T_i, U_i, X_i, Y_i, \Lambda_i = \mathbb{1}_{\mathbb{G}_\mathbb{T}}) \in (\mathbb{G}_1 \times \mathbb{G}_2)^2 \times \mathbb{G}_T$. We denote the level of tag as $lev(\sigma_i)$ and we set $lev(\sigma_i) = 1$.
   - Set $\sigma_x = (\Delta, \mu_1, \sigma_1, \ldots, \mu_n, \sigma_n)$ and $\tau_x = \perp$.

3. **Compute**$_{PK}(\sigma_x) \rightarrow \sigma_y$ with an admissible circuit f it consist of the following steps:
   - Run the evaluation circuit f on the BGV encrypted data to obtain $\mu =$ BGV.Eval$(f, \mu_1, \ldots, \mu_n)$.
   - Apply (gate-by-gate) f over the authentication tags $(\sigma_1, \ldots, \sigma_n)$, using the following gate functions **GateEval()**.
     GateEval:$(f_g, \sigma_1, \sigma_2) \rightarrow \sigma$
     Parse $(T_i, U_i, X_i, Y_i, \Lambda_i) \in (\mathbb{G}_1 \times \mathbb{G}_2)^2 \times \mathbb{G}_T$ for $i = 1, 2$, where f_g stands for tag addition ("+") or tag multiplication ("×"). We try to compute $\sigma = (T, U, X, Y, \Lambda)$.
     - Add two tags together. If f_g ="+", the addition takes different forms depending on the levels of the input tags.
       If $lev(\sigma_1) = lev(\sigma_2)$, then $\sigma = (T_1 \cdot T_2, U_1 \cdot U_2, X_1 \cdot X_2, Y_1 \cdot Y_2, \Lambda_1 \cdot \Lambda_2)$ with $lev(\sigma) = 1$.
       Else, without loss of generality, let suppose that $lev(\sigma_1) = 1$ and $lev(\sigma_2) = 2$(i.e. there is a multiplication gate before this gate).
       The idea is to create a level-2 tag (σ_1') from σ_1 as follows: $\sigma_1' = (e(T_1, h), e(g, U_1), e(X_1, h), e(g, Y_1), \Lambda_1)$. Then compute $\sigma = \sigma_1' + \sigma_2$ as in the first case but set $lev(\sigma) = 2$.
     - Add a constant to a tag $(c + \sigma_1)$. This method depends on the level of the tag as follows:
       If $lev(\sigma_1) = 1$, then the result tag $\sigma = (T_1 \cdot (g^c), U_1 \cdot (h^c), X_1, Y_1, , \Lambda_1)$.
       If $lev(\sigma_1) = 2$, then we obtain: $\sigma = (T_1 \cdot (e(g,h)^c), U_1 \cdot (e(g,h)^c), X_1, Y_1, Z_1, \Lambda_1)$.
       In both cases $lev(\sigma) = lev(\sigma_1)$.
     - Multiplication by a constant $(c \cdot \sigma_1)$. The result tag is $\sigma = (T_1^c, U_1^c, X_1^c, Y_1^c, \Lambda_1^c)$ and $lev(\sigma) = lev(\sigma_1)$.
     - Multiplication. For f_g ="×" on two tags $(\sigma_1 \times \sigma_2)$
       If $lev_y(\sigma_1) > 1$ or $lev_y(\sigma_2) > 1$ then reject. Else calculate $T = e(T_1, U_2)$, $U = e(T_2, U_1)$, $X = e(X_1, U_2) \cdot e(X_2, U_1)$, $Y = e(T_2, Y_1) \cdot e(T_1, Y_2)$, $\Lambda = e(X_1, Y_2)$. Also set $lev(\sigma) = 2$.
       Its not necessary to keep U and Y after a multiplication because $T = U$ and $X = Y$. We keep them only for the sake of clarity. As noted in [3],

one can see the function f as the composition of two functions $f_g(f_1, f_2)$ in the last gate f_g of f.

– Set $\sigma_y = (\Delta, \mu, \sigma)$, where σ is the tag obtained after evaluating the last tag of f.

4. **Verify**$_{SK}(\sigma_y, \tau_x) \rightarrow (acc, y)$, for $\sigma_y = (\Delta, \mu, \sigma)$, using the following operations:

- Compute $\hat{H}_\kappa(\mu) \rightarrow \hat{\nu}$.
- Run $\text{CFEval}^{on}_{\Delta}(K, w_f) \rightarrow W$
- Check, depending on the of degree of f, as follows:

(a) If $deg(f) = 1$, check the following equations:

$$(T, U) = \hat{\nu}(= (g^{((\mu(\alpha))(\beta))}, h^{((\mu(\alpha))(\beta))}))$$
$$e(X, h) = e(g, Y)$$
$$W = e(T \cdot X^a, h).$$

(b) Else, check over $\mathbb{G}_T$ the following equations:

$$T = U = \hat{\nu}(\text{ i.e. } = e(g, h)^{((\mu(\alpha))(\beta))}) \tag{4.5}$$
$$X = Y \tag{4.6}$$
$$W = T \cdot (X)^r \cdot (\Lambda)^2 \tag{4.7}$$

- If all equations are satisfied set the check bit acc to 1 (accept), otherwise set it to 0 (reject).
- Finally, if acc=1, $\mu' = \mu \ mod \ \Phi_m(x) = (c_0, c_1, c_2)$ and set $y = BGV.Dec_{dk}(\mu')$, otherwise set $y = \perp$.

Theorem 5. [3] If BGV is a semantically secure homomorphic encryption scheme, $\hat{\text{H}}$ is a collision-resistant homomorphic hash function and **F** is a pseudorandom function, then VC described above is correct, adaptive secure and input private.

4.4.2 VC for Paillier scheme

Now, we present the VC for Paillier cryptosystem [129]. More precisely, its a Linearly Homomorphic Authenticated Encryption scheme with Public Verifiability (LAEPuV) and provable correctness called LEPCoV and it allows the public verifiability of data returned by the server. This scheme improves Catalano et al.'s instantiated scheme [130] by avoiding false negatives during the verification step.

Let $S = (KeyGenS, Sign, Verify)$ be a signature scheme.

- $AkeyGen(sz, I)$: takes as input a prime size sz (in number of bits) and an integer I representing the upper bound for the number of messages encrypted in each dataset. It calculates the secure sk and public pk parameters as follows: sample four (safe) primes p_E, q_E, p_S, q_S of size $sz/2$, such that $N_E = p_E \cdot q_E$ and $N_S = p_S \cdot q_S$ it holds that $\varphi(N_S) = (p_S - 1)(q_S - 1)$, the group elements g_0, g_1, h_1,..., $h_I \in \mathbb{Z}^*_{N_s}$ and $g \in \mathbb{Z}^*_{N_E^2}$ of order N_E, and picks a hash function H. Finally, it runs $KeyGenS(sz)$ to

obtain the secure and private signature key (sk_S, pk_S). Returns the key pair (sk, pk), where $pk = (N_E, g, N_S, g_0, g_1, h_1, \ldots, h_I, H, pk_S)$ and $sk = (p_E, q_E, p_S, q_S, sk_S)$.

- $AEncrypt(sk, \tau, i, m)$: probabilistic algorithm that takes as input the secret parameter, a message $m \in M$, a dataset identifier τ, and an index $i \in \{1, \ldots, I\}$ to calculate the ciphertext c containing the encryption of the message with the tag of verification. Thus, it computes the Paillier encryption C of the message m, $R = H(\tau||i)$, and a tuple $(a, b) \in \mathbb{Z}_{N_E} \times \mathbb{Z}^*_{N_E}$such that $g^a b^{N_E} = CR \mod N_E^2$ (using the factorisation of N_E). In addition, if τ is used for the first time, it chooses a not yet used prime e of length $l \leq sz/2$ such that $gcd(eN_E, \varphi(N_S)) = 1$, it computes its inverse e^{-1} mod $\varphi(N_S)$, and its signature $\mu_e = Sign_{sk_S}(\tau||e)$ and it stores (τ, e, e^{-1}, μ_e) in the list L. Otherwise, it takes (τ, e, e^{-1}, μ_e) from the list L. Then, it chooses an element s uniformly at random from $\mathbb{Z}_{eN_E}$ and it computes x using the p_S and q_S such that $x^{eN_E} = g_0^s h_i g_1^a \mod N_S$. It returns $c = (C, e_i, e_i^{-1}, \mu_e, \tau, \sigma)$, where $\sigma = (a, b, s, x)$ is the verification tag.

- $AEval(pk, \tau, f, \{c_i\}_{i\in[I]})$: takes as input the public key pk, a dataset identifier τ, a linear function $f = (f_i)_{i\in I}$ and I ciphers $\{c_i\}_{i\in[I]} = (C_i, e_i, e_i^{-1}, \tau_i, \sigma_i)$. The output is a cipher c. The algorithm checks if there exists an index $l{\in}[I]$ such that $\tau{\neq}\tau_l$, or that the signature $(Verify(pk_S, \tau||e_l, \mu_{e_l}) = 0)$. Furthermore, the algorithm checks if there are two indexes $i \neq j \in [I]$ such that $e_i \neq e_j$. If one of the checks is true, the algorithm aborts. Otherwise, the algorithm sets $e = e_1, e^{-1} = e_1^{-1}$, $\mu_e = \mu_{e_1}$ and evaluates f over ciphertext as: $C = \prod_{i=1}^{I} C_i^{f_i} \mod N_E^2$. It also evaluates f over the tag to obtain a new tag (a, b, s, x) as follows: $a = \sum_{i=1}^{I} f_i a_i \mod N_E$, $b = \prod_{i=1}^{I} b_i^{f_i} \mod N_E^2$, $s = \sum_{i=1}^{I} f_i s_i \mod eN_E$, $s' = \left(\sum_{i=1}^{I} f_i s_i - s\right)/(eN_E)$, $a' = \left(\sum_{i=1}^{I} f_i a_i - a\right)/N_E$, and $x = \frac{\prod_{i=1}^{I} x_i^{f_i}}{g_0^{s'} g_1^{a' e^{-1}}} \mod N_S$.
It returns the cipher $c = (C, e, e^{-1}, \mu_e, \sigma)$.

- $AVerify(pk, \tau, c, f)$: takes as input the public key pk, a dataset identifier τ, a cipher $c = (C, a, b, e, s, \tau, x)$, and a linear function $f = (f_i)_{i\in[I]}$, to detect if c is a valid or invalid cipher. For this goal, the algorithm checks that:

$$\begin{cases} Verify(pk_S, \tau||e, \sigma_e) = 1; \\ a, s \in \mathbb{Z}_{eN_E}; \\ x^{eN_E} = g_0^s \prod_{i=1}^{I} h_i^{f_i} g_1^a \mod N_S; \\ g^a b^{N_E} = C \prod_{i=1}^{I} H(\tau||i)^{f_i} \mod N_E^2; \end{cases}$$

  If all checks pass, it outputs 1 (i.e c is a valid cipher), else it outputs 0, (i.e. c is an invalid cipher).

- $ADecrypt(sk, \tau, c, f)$: Taking as input the secret parameter sk, a data set identifier, a cipher c, and a linear function $f = (f_i)_{i\in[I]}$, it calculates the decryption of c or $\perp$ (if c is invalid cipher). Then it verifies if c is a valid cipher by running $AVerify$ (pk, τ, c, f). If passed, the algorithm returns the message m obtained by $Dec(c)$. Otherwise, it returns $\perp$.

We refer the reader to [130] for the proofs of correctness and security for the LAEPuV scheme and to [129] for the security and correctness proofs of the LEPCoV scheme.

Part II

Our contribution

Chapter 5

Computing NN using VC and FHE

5.1 Introduction

Despite limitations due to high communication overheads, computing costs or expressivity, techniques for computing over encrypted data such as Fully Homomorphic (FHE), Functional Encryption (FE) or Multi Party Computation (MPC), to name a few, are becoming practical for a number of applications. At the same time, Artificial Intelligence (AI) techniques and, especially, Neural Networks ones are becoming omnipresent in our connected society and have already lead to countless practical applications impacting, for better or worse, our daily lives. Yet, the AI applications ecosystem has so far developed with a limited concern for user privacy.

In this context, we contribute to the study of how the aforementioned emerging cryptographic techniques can contribute to address AI privacy challenges. More specifically, we address the issue of ensuring the end-to-end confidentiality of some *user* data when they pass through a neural network operated on some cloud *server* with the aim of providing classification results to an *operator*. We do so by means of Homomorphic Encryption,

which is used to execute the neural network on the server, associated to Verifiable Computing techniques in order to guarantee both the *confidentiality* of the user data as well as the *integrity* of the execution of the network with respect to threats coming from the server. Still, this is easier said than done, and in order to cope with the various constraints coming when using these techniques we also have to open the machine learning box and to propose a specific solution in which both the neural net structure and the crypto techniques are co-designed in order to achieve the desired overall system security properties.

5.1.1 Problem statement and contribution

We propose an approach to build privacy preserving neural networks, combining a FHE cryptosystem (here the BFV scheme [24]) with the Verifiable Computation protocol from Fiore et al. [3], adapted for BFV encrypted data.

More specifically, we show how to evaluate the first layer of a neural network on homomorphically encrypted data on a server and how the operator, which decrypts them, can check the result validity. The operator then pursue, in the clear domain, the network evaluation to reach a final classification.

We present here a global architecture made of these three entities: the client, owner of some private data, the server performing the computation over the encrypted layers of the neural network and the operator computing the remaining layers of the neural network. This architecture allows to deploy our semi-private evaluation of a neural network in order to ensure data privacy for clients and also provide integrity proofs with respect to the server computations. A complete description of the proposed architecture and an analysis of the associated security threats are given in Section 5.2.

To achieve this we reuse the neural network model proposed in [4] in which the first layer of the network acts as a whitener, ensuring, by means of adversarial training, that the knowledge of the outputs of the first layer does not allow to recover selected (sensitive) features of the input data while still preserving an ability to perform good quality classification. However, as shown in section 5.3, the implementation of our approach targets different use cases and deployment scenarios as the ones from [4]. Although based on different reference problems, since the underlying cryptographic primitives we use are different (i.e. HE and VC), our work has similar security guarantees. This work thus complements [4] approach by providing more versatility to their network partitioning approach. Also note that the neural network design, partitioning and training approaches underlying [4] is widely applicable to virtually any classification problem.

We also provide an efficient implementation and demonstrate practical results on the MNIST dataset [131], for the recognition of handwritten digits. Within our approach we perform the classification of the encrypted image in less than 3.8 seconds and the integrity check for the evaluation of the first layer in approx. 0.015 sec, with an overall accuracy of 97.54% for 128 bits of security.

5.2 Related work

5.2.1 FHE for encrypted machine learning.

Research on the application of techniques for computing over encrypted data, FHE or others "competing" techniques, to neural networks-related privacy issues is only at its beginning. The first attempts at applying homomorphic encryption techniques to neural networks have almost all focused on the inference phase and more specifically, as the present work does, on the problem of evaluating a public (from the point of view of the computer doing the evaluation) network over an encrypted input (hence producing an encrypted output). The first work of this kind is CryptoNets [132] where the authors successfully evaluate an approximation of a simple 6-layer Convolutional NN able to achieve 99% success recognition on the well-known MNIST hand-written digits database. That network was composed only of simple Convolution, Square Activation and Mean Pool Scaling layers with only one application of the "FHE-unfriendly" Sigmoid function at the last layer which, in that specific case, could be dropped without affecting prediction quality (hence the final network only had 2 nonlinear Square Activation layers leading to a small multiplicative depth). Their implementation uses the BFV FHE scheme [24] and achieves network evaluation timings of around 4 minutes on a high-end PC. Yet, thanks to the SIMD/batching property of FV-like schemes, one network evaluation can in fact lead to 4096 evaluations of the network done in parallel on independent inputs (i.e. the network is evaluated once on ciphertexts which have many "slots" and thus contain different cleartexts). So, although the latency remains of 4 minutes, the authors rightfully claim their system to be able to sustain a throughput of around 60000 digit recognitions per hour. In subsequent papers, Chabanne et al. [133], [134] are building approximations with small multiplicative depth of networks with up to 6 nonlinear layers by combining batch normalization layers with degree 2 approximations of the ReLU function (the former allowing to stabilize the inputs of the latter in order to decrease the sensitivity of the network to approximation errors). Through significant hand-tuning of the learning step of their networks, they show that these can achieve state-of-the-art prediction quality on both hand-digit (MNIST) and face recognition. However their work lacks an implementation and, hence, they did not provide FHE evaluation timings. More recently, Bourse et al. [135], have fine-tuned the TFHE cryptosystem towards a slight generalization of BNNs (Binary Neural Networks) called DiNNs in which the nonlinear activation function is the sign function which they intricate with the TFHE bootstrapping procedure for more efficiency. Overall, they are able to evaluate small DiNN networks (100 neurons and only one hidden layer) in around 1.5 seconds resulting in a (just decent) 96% prediction accuracy on the MNIST database. This line of research has also been pursued in [136] where the authors have fine-tuned the TFHE cryptosystems for efficient evaluation of Hopfield networks and tested their approach on a face recognition application achieving the evaluation of an encrypted network (with 256 neurons) over an encrypted input in 0.6 secs, however for a recognition accuracy of only 86%. This latter work is the first to attempt at both hiding the network and its input (and, by construction, its output). Also, in [137], the authors focus on applying FHE to hide the model of a neural network-based system in the case of a plain input for the special case of embedding-based networks.

Other notable works on the application of homomorphic encryption techniques to the private inference step of ANN include, in a non-exhaustive way, nGraph-HE [138], nGraph-HE2 [139], LOLA [140], TAPAS [141], NED [142], Faster CryptoNets [143]. As already

emphasized, all the previously mentioned papers focus only on the inference phase under the hypothesis of an honest-but-curious evaluation server. It should also be mentioned that the applications to ANN of other "competing" techniques for computing over encrypted data, the main one being Secure Multiparty Computations (MPC) also start to be investigated in their associated communities (e.g., [144], [145]).

5.2.1.1 VC for machine learning.

As for applying the verifiable computing protocols for the computation of Neural Networks, there are only a few recent works on this subject.

The SafetyNets [146] is an interactive proof protocol to execute a deep neural network on a cloud, using Interactive Proof Systems [98] to prove the correctness of the calculated result returned by the cloud server. As such, it requires multiple interactions and calculations with the server to complete the verification step and it replaces the ReLu function by the function $x \to x^2$, which reduces the neural network accuracy. Since it is impossible for the prover to prove a non-deterministic computation (i.e. to prove the correctness of a computation while hiding some inputs) then the verifier and the prover need to share the model. Zaho et al. [147] propose VeriML to verify a neural network using QAP-based zk-SNARK. The VeriML ensures both security statement (privacy and integrity) but, it has a fairly large proof complexity $O(|\vec{a}| \cdot |\vec{x}| + |\vec{y}|)$ where $\vec{a}$ denotes the kernel, $\vec{x}$ the input and $\vec{y}$ the output. The combination of the GKR protocol and QAP (Quadratic Arithmetic Programs) scheme has been proposed by Chabanne et al. [148]. To do this the verifying process of GKR is verified in the QAP circuit. However; this still leads to a large computation complexity of $O(|\vec{a}| \cdot |\vec{x}| + |\vec{y}|)$ (according to [149]).

In the same line of research, Keuffer et al. build [150] a Verifiable Computing scheme, by combining other two VC schemes: a general-purpose VC (GVC) like [52, 151] and a specialized one (EVC), namely Sum-Check protocol [152]. As such, they achieve efficient verification of complex operations as for example for large matrix-multiplication. In order to verify a function, they perform the complex operation with the EVC protocol where the GVC is least efficient, the remaining functions being handled by the GVC. They apply this VC scheme to prove the correctness of a neural network evaluation.

As seen in this section and to the best of our knowledge, so far there are no approaches for an outsourced machine learning method which support the integrity of its execution while guaranteeing the data privacy by means of verifiable computation and homomorphic encryption.

5.2.2 Encrypted machine learning using Functional Encryption

As already emphasized, our model for the neural network builds on the one from [4] in which it is used for a partial encrypted-domain network evaluation using Functional Encryption. Let us describe it here more in details.

To evaluate the first layer of the neural network, they use the Functional Encryption (FE) scheme, from [153], designed for quadratic multi-variate polynomials, based on bilinear pairings and with adaptive security under chosen-plaintext attacks (IND-CPA security). As illustrated in Figure 5.1, the method they propose achieves user data privacy when performing classification in a classical user-operator model. It is based on a neural network

composed of a private (running in the encrypted domain) and a public (running in the clear domain) part, with the private part consisting of a quadratic evaluation function. In their approach, the user encrypts his/her data x using a FE public key pk and sends the encryption $Enc_{FE}(x)$ to the operator.

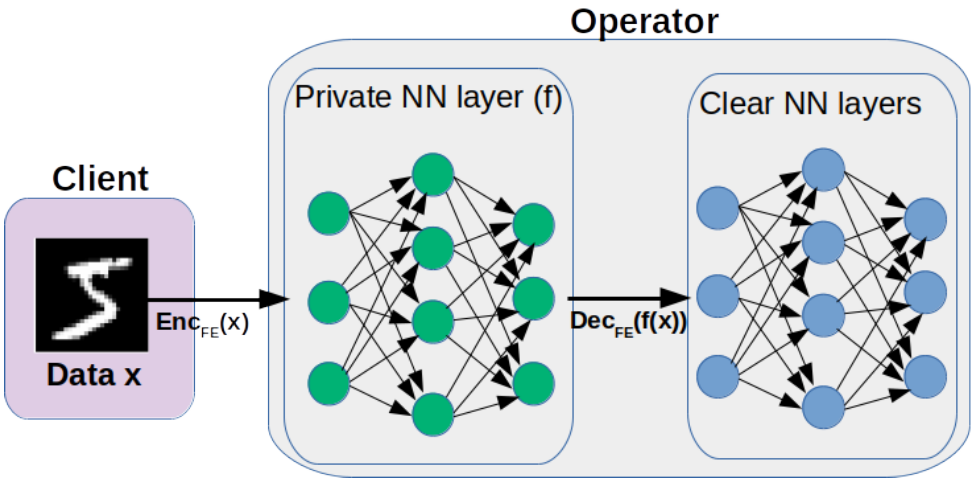

Figure 5.1: Semi-encrypted Neural Network with Functional encryption

To classify, the operator applies the first layer of the neural network over the data it received. More formally, the operator runs the quadratic activation function f over encrypted data by means of the FE scheme and uses the decryption key dk_f to decrypt the result of this layer as plaintext. These decrypted results are injected in the remaining of the neural network which is then evaluated on clear data with the argmax function applied at the end to obtain the cleartext output. They run the overall neural network on top of a modified version of MNIST where there are two types of classes to predict: the public label which is the digit on the image and the private label associated with the font used to draw the image.

Moreover, they also propose a counter-measure to the threat associated from collateral learning, coming from an adversary having access to the output of the quadratic network and wanting to learn the private label (e.g. the font). As such, in order to reduce the information leakage on the operator hosting the partially encrypted neural network, they employ a semi-adversarial training method.

In this contribution, we choose to build upon their quadratic model for the first layer as well as the same remaining neural network. However, our work is different in several aspects. First at all, even if the architectural framework is very similar our approach targets different deployment scenarios and use cases (see below). Second, we ensure the data privacy and security using BFV homomorphic encryption scheme as well as a VC protocol, adapted from [3] so we use different underlying cryptographic primitives. Finally, as shown in the experimental part, in terms of performances, we obtain similar and sometimes even better execution times for the different steps of the private classification.

In essence, the two approaches are complementary: in the Functional Encryption approach, there is also a server playing the role of a trusted authority for the generation and distribution of the keys but which does not perform any calculations. Therefore, it has only an offline key management role i.e., it only has to provide (once) the master public key to the user as well as the secret functional decryption key associated to the

first network layer to the operator and plays no role in the processing of a client request. In our setting, the server has an online active role in the sense that it is the entity receiving the encrypted client data and evaluating the first network layer (in the encrypted domain). Thanks to the use of VC in our approach which provides the server with the execution integrity which FHE alone does not provide, both approaches are equivalent in terms of security model. They differ on where the main computing burden (primarily due to the encrypted data processing) occurs: on the operator in the Functional Encryption approach or on the server in our FHE/VC approach. It is difficult to state which of the two is more relevant in practice as it clearly depends on the use-case at hand and where some computational power is most naturally available (e.g., if the operator is a mobile device such as a tablet then our approach is more relevant whereas in other cases it may not be so).

5.3 Scenario and threat model

This section provides a general scenario of deployment for our method, the different threats we address as well as the possible use cases.

We start by describing the general architecture in which we apply a neural network for a semi-private evaluation of a user data. There are mainly three entities involved: the user/-querier, owner of some confidential data x, the server performing the privacy-preserving part of the neural network and an operator having access to the evaluation of this preliminary classification and performing the remaining of the neural network in the clear domain.

As such, the server evaluates in the homomorphic domain the first layer of the neural network over the private data of one or many users. In our approach, this private step is equivalent to the homomorphic evaluation of a quadratic function (which is totally feasible and moreover with really good performances by existing FHE means).

Unlike other works using homomorphic encryption for private inference, we set up our study in the case of a *malicious* server, which can possibly alter the results of the evaluation (e.g. by not running the specified algorithm). To counter this, within our setting, the server has to generate an integrity proof aside of the homomorphic results without any interaction with the user or the operator. To do so, we make use of the VC protocol of Fiore et al. [3] which allows to efficiently check that a computation over encrypted data has been properly performed. To the best of our knowledge, this scheme is presently the most practical to address the validity of computation over encrypted data with the evaluation of multi-variate polynomials of degree at most 2. As VC schemes for degree beyond 2 are not practical, this is one of the reasons we restrain the homomorphic evaluation to a first quadratic layer (practical Functional Encryption scheme also have the same limitations[1]).

Then, the homomorphic evaluation of the private neural network along with the associated integrity proof is sent to the operator. The last one decides (based on the proof) if the server output is correct and, when it is so, can decide to decrypt the homomorphic results of this first layer and to continue with the prediction on clear data. As a counter-

[1]This is due to the need to go beyond bilinear maps to achieve higher degrees in the underlying cryptographics primitives involved in both VC and FE.

measure in the case of an operator which takes advantage of the decrypted results of this intermediate layer operated by the server to recover the user sensitive data, the quadratic first layer is trained based on the adversarial learning technique from [4]. Let us also note that, since it is performed on plaintext data, this remaining part of the network can involve more complex machinery and obviously more than one additional layers (including non linear activation functions).

In summary, under the hypothesis of non collusion between the entities involved, the architecture we propose has the following security properties:

- The user has access (obviously) to its own data x and, in function of the use case, can have access to the overall classification result or the evaluation of the intermediate first layer (if the operator shares it).
- The server, evaluating the first private layer of the neural network, has no access to the inputs x nor to the output of the function $f(x)$. While the homomorphic encryption addresses the confidentiality threats on the user inputs, the verifiable computation addresses the integrity threats to the homomorphic evaluation, in the case of a malicious server.
- The operator, performing the remaining layers of the neural network, has access to the decryption result of the first layer and can check the validity of the server computation. It can then exploit the overall result of the evaluation of the neural network to its own advantage or return it to the user. The adversarial training model for the first layer addresses the case of an honest-but-curious operator which may try to learn sensitive information about the user inputs based on the intermediate results.

Table 5.1 illustrates the threat analysis in terms of the access to the sensitive data x and to the result of the evaluation of the function f over x for all three entities involved in our architecture.

	User	*Server*	*Operator*
x	Y	N	N
$Enc(x)$	Y	Y	N
$f(x)$	N	N	Y
$Enc(f(x))$	N	Y	Y

Table 5.1: Threat analysis in our architecture (Y: Yes, he has access; N: Non, he has no access)

Let us know illustrate some applications in which our architecture could be useful.

Mail filter. In this application, we consider the scenario were an employer (operator) wishes to perform statistics on its employees (users) emails. For example, she wishes to know when an email is received whether it is professional or personal, a phishing attempt, some advertisement or some spam. The employer needs to do so without having access to the employees mail contents. In this context, a cloud provider can play the role of the server. First, the employees encrypt their emails under the employer's public key and forward them to the cloud provider. The cloud provider then runs the first neural net layer in the encrypted domain and sends the results to the employer (along with the

proof that the first layer was applied correctly) which then turns them into a concrete classification to compute her statistics. In this setup, employees have to trust only that the cloud server will not collude with their employer (e.g. by forwarding its encrypted emails). In particular, the confidentiality of their emails is safe from server threats thanks to the FHE layer. Thanks to VC protocol, the employer is guaranteed that the cloud provider evaluates properly the first layer of her network. Also, because it reveals only the first layer of its networks, the employer does not have to disclose the exact statistics she in fine computes to the cloud provider.

Medical use-case. Suppose that a pharmaceutical firm wishes to conduct an epidemiologic study over a group of people. To do so, they need to evaluate for example a specific neural network on some health-related data over a large set of patients while respecting the following properties: (1) the evaluation of the NN should not be done by the patients (i.e. for either or both cost and intellectual property issues) and (2) it needs to access only the outputs of the neural net and it is not allowed to access the inputs of this network. For this goal, consider a trusted health authority (server) in the center between these patients (clients) and the firm (operator). The firm is the owner of all keys in our architecture (FHE and VC keys). To apply its network, the firm (the operator in our architecture) discloses its first network layer to the health authority. It is the authority responsibility to guarantee that the first layer is acceptable in terms of privacy of the input data (one nice thing is that the firm discloses only the first layer of its network to the authority) i.e. that knowledge of the outputs of the first layer does not allow to recover specific features of the associated inputs (after decryption of these outputs by the operator). If the authority validates the first layer, it distributes the firm's public key to the patients, which they use to encrypt their data which are then sent to the authority. The authority then evaluates the first layer of the neural network in the FHE domain and then sends its (encrypted) results along with short proof of correctness to the firm. Finally, the firm uses its secret key with the short proof it received, to verify the calculation of the server. Then if the verification is successful, the firm decrypts the result and evaluates the remainder of its network performed on clear data. The security properties are specified as above.

5.4 Technical preliminaries

5.4.1 FHE

Fully Homomorphic Encryption (FHE) schemes allow to perform arbitrary computations directly over encrypted data. That is, with a fully homomorphic encryption scheme E, we can compute $\mathsf{E}(\mathsf{m}_1 + \mathsf{m}_2)$ and $\mathsf{E}(\mathsf{m}_1 \times \mathsf{m}_2)$ from encrypted messages $\mathsf{E}(\mathsf{m}_1)$ and $\mathsf{E}(\mathsf{m}_2)$.

In this section we recall the general principles of the **BFV** homomorphic cryptosystem [24], which we use in combination with a $\mathcal{VC}$ scheme. Since we know in advance the function to be evaluated homomorphically, we can restrain to the somewhat homomorphic version described below. Moreover, we skip the description of the relinearisation step not needed in our approach which evaluates only multi-variate quadratic polynomials.

Let $R = \mathbb{Z}[x]/\Phi_m(x)$ denote the polynomial ring modulo the m-cyclotomic polynomial with $n' = \varphi(m)$. The ciphertexts in the scheme are elements of polynomial ring R_q, where R_q is the set of polynomials in R with coefficients in $\mathbb{Z}_q$. The plaintexts are polynomials

belonging to the ring $R_t = R/tR$.

As such, **BFV** scheme is defined by the following probabilistic polynomial-time algorithms:

BFV.ParamGen(λ): $\rightarrow (n', q, t, \chi_{key}, \chi_{err}, w)$.
It uses the security parameter λ to fix several other parameters such as n', the degree of the polynomials, the ciphertext modulus q, the plaintext modulus t, the error distributions, etc.

BFV.KeyGen$(n', q, t, \chi_{key}, \chi_{err}, w) :\rightarrow (pk, sk, evk)$.
Taking as input the parameters generated in **BFV.ParamGen**, it calculates the private, public and evaluation key. Besides the public and the private keys, an evaluation key is generated to be used during computation on ciphertexts in order to reduce the noise.

BFV.Enc$_{pk}(m) \rightarrow c = (c_0, c_1, c_2 = 0)$
It produces a ciphertext c according to BFV-cryptosystem for a plaintext m using the public key pk.

BFV.Dec$_{sk}(c) :\rightarrow m$
It computes the plaintext m from the ciphertext c, using private key sk.

BFV.Eval$_{pk,evk}(f, c_1, \ldots, c_n) :\rightarrow$ c, with $c =$**BFV.Enc**$_{pk}(f(m_1, \ldots, m_n))$, where $c_i =$ **BFV.Enc**$_{pk}(m_i)$, and f has n inputs and has degree at most two.
It allows the homomorphic evaluation of f, gate-by-gate over c_i using the following functions: **BFV.Add**(c_1, c_2) and **BFV.Mul**$_{evk}(c_1, c_2)$.

For further details on this scheme, we refer the reader to the paper [24].

Let us just note that a BFV ciphertext c can be seen as an element in $R_q[y]=$ $\mathbb{Z}/q\mathbb{Z}[X, Y]/\Phi_m(x)$ with a degree at most 2 (i.e., $c = c_0 + c_1 y + c_2 y^2$).

5.4.2 VC

Verifiable computation $\mathcal{VC}$ techniques allow to prove and verify the integrity of computations on authenticated data. A Verifiable Computation scheme is defined as a protocol in which a client (usually weak) has a function f and some data denoted x and delegates to another client (in most cases a server) the computation of $y = f(x)$. Then the same client or another one can receive the result y plus a short proof of its correctness. More in details, a user generates an authentication tag σ_x associated with his/her data x with his/her secret key and the server computes an authentication tag $\sigma_{f,y}$ that certifies the value $y = f(x)$ as an output of the function f. Now, anyone using the verification key (public or secret) can verify y to check that y is indeed the result of $f(x)$.

A $\mathcal{VC}$ scheme includes the following algorithms:

1. (PK,SK)$\leftarrow$**KeyGen**(f, λ): Taking as input the security parameter λ and a function f, this randomized key generation algorithm generates a public key (that encodes the target function f) used by the server to compute f. It also computes a matching secret key, kept private by the client.

2. $(\sigma_x, \tau_x) \leftarrow$**ProbGen**$_{SK}(x)$: The problem generation algorithm uses the secret key SK to encode the input x as a public value σ_x, given to the server to compute with, and a secret value τ_x which is kept private by the client.

3. $\sigma_y \leftarrow$**Compute**$_{PK}(\sigma_x)$: Using the client's public key and the encoded input, the server computes an encoded version for the function output $y = f(x)$.

4. $(acc, y) \leftarrow$**Verify**$_{SK}(\tau_x, \sigma_y)$: Using the secret key SK and the secret τ_x, this algorithm converts the server output into a bit acc and a string y. If $acc = 1$ we say that the client accepts $y = f(x)$, meaning that the proof is correct, else (i.e. $acc = 0$) we say the client rejects it.

5.4.3 Pseudo Random Function with Amortized closed-form Efficient

We present here the notion of Pseudo Random Function(**PRF**) with Amortized Closed-Form Efficiency [55].

A **PRF** consists of two algorithms **(F.KG, F)**. The key generation method **F.KG** takes as input the security parameter λ to generate a secret key K and some public parameters pp that specify the domain χ and the range $\mathcal{R}$ of the function F. The function F_K takes as input the data $x \in \chi$ and uses the key K to generate a value $R \in \mathcal{R}$ satisfying the following pseudorandom property:

Definition 16. [55] Consider a computation **Comp** that takes as input n random values $R_1, \ldots, R_n \in \mathcal{R}$, and a vector of m arbitrary values $z = (z_1, \ldots, z_m)$, and assume that the computation of **Comp**$(R_1, \ldots, R_n, z_1, \ldots, z_m)$ requires time $t(n, m)$. Let $L = (L_1, \ldots, L_n)$ be arbitrary values in the domain χ of **F** such that each one can be interpreted as $L_i = (\Delta, \tau_i)$. We say that a PRF (KG, F) satisfies amortized closed-form efficiency for $(Comp, L)$ if there exist two algorithms $\mathbf{CFEval}^{off}_{Comp,\tau}$ and $\mathbf{CFEval}^{on}_{Comp,\Delta}$ such that:
1. Given $w \leftarrow CFEval^{off}_{Comp,\tau}(K, z)$ we have that:

$CFEval^{on}_{Comp,\Delta}(K, w) = Comp(F_K(\Delta, \tau_1), \ldots, F_K(\Delta, \tau_p), z_1, \ldots, z_m)$
2. the running time of $CFEval^{on}_{Comp,\Delta}(K, w)$ is $o(t)$.

5.4.4 Homomorphic Hash function

Informally, a family of key homomorphic hash functions **H** with domain $\mathcal{X}$ and range $\mathcal{R}$ consists of three algorithms (**H.KeyGen, H, H.Eval**). The first one, the key generation hash **H.KeyGen**, generates the description of the hash function H_K,where K is the key, the function **H** computes the hash and, finally, **H.Eval** allows the computation over $\mathcal{R}$ satisfying the following homomorphic property: $H.Eval(f, (H(x_1), \ldots, H(x_n))) = H(f(x_1, \ldots, x_n))$ where $x_i \in \mathcal{X}$ (**H** is a ring homomorphism).

In the $\mathcal{VC}$ scheme for quadratic multi-variate polynomial over BFV encrypted data, we are interested in the calculation of **H.Eval** for one level of multiplication (with two inputs) and any numbers of additions over $\mathcal{D}$.

In chapter 4, we presented the realization of homomorphic hash $\hat{\mathbf{H}}$, based on bilinear groups. It allows to reduce a **BFV** ciphertext $\mu \in R_q[y]$ into a $v \in \mathbb{Z}/q\mathbb{Z}$ depending on the degree of μ denoted $deg_y(\mu)$ with preservation of the homomorphic properties. Hence $H.Eval(f, (H(\mu_1), \ldots, H(\mu_n))) = H(f(m_1, \ldots, m_n))$ where $\mu_i =$ **BFV.Enc**$_{PK}(m_i)$ with m_i a BFV plaintext.

5.5 VC for Quadratic polynomials over BFV Encrypted data

In this section, we present an application of the VC scheme of Fiore et al. [3] for the case of multi-variate polynomials of degree 2, over **BFV** encrypted data instead over **BGV** encrypted data as in the original paper. First the client encrypts his/her data $x = (x_1, \ldots, x_n)$ as a **BFV** ciphertext, where the plaintext modulus q is chosen to be prime. In parallel with the encryption of x_i, he/she also generates a tag σ_i for his/her data, using the combination of the PRF output and the hash collision-resistant functions (that compresses a **BFV** ciphertext into a double of group elements). Once the server receives the BFV ciphertexts $[x_i]_{BFV}$ and the tags σ_i from the user, it computes f over $[x_i]_{BFV}$ and over σ_i and it obtains $y = f([x_i]_{BFV})$ and respectively a tag $\sigma = f(\sigma_i)$. The server sends y with the associated tag to the user owning the verification keys, which then checks the output in constant time (because he has already done a pre-computation phase).

In this scheme, we require to authenticate with our scheme:

1. Each of the $2n' - \mathbb{F}_q$ components of a BFV ciphertext.
2. The BFV evaluation circuit $\hat{f} : \mathbb{F}_q^{2nn'} \to \mathbb{F}_q^{3n'}$ instead of $f : \mathbb{F}_q^n \to \mathbb{F}_q$.

More formally, our $\mathcal{VC}$ scheme is specified by the following algorithms:

1. **KeyGen**$(f, \lambda) \to (PK, SK)$, with the following steps:
   - First generate $bgpp = (q, g, h, e)$ some bilinear group parameters, where $\mathbb{G}_1 =< g >$, $\mathbb{G}_2 =< h >$, $q = order(\mathbb{G}_i)$ for $i = \{1, 2\}$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a non-degenerate bilinear map ($\mathbb{G}_T =< e(g, h) >$).
   - Run **BFV.ParamGen**$(\lambda) \to (n', q, t, \chi_{err}, \chi_{key}, w)$ to generate the parameters for the **BFV** encryption scheme. Run **BFV.KeyGen()**$\to (pk, sk, evk)$.
   - Run $\tilde{H}.KeyGen \to (\kappa, \tilde{K})$ to choose a random member of the hash function family $\tilde{H} : \mathcal{D} = \{\mu \in \mathbb{Z}_q[x][y] : deg_x(\mu) \leq 2(n' - 1)\ , deg_y(\mu) \leq 2\} \subset R_q[y] \to \mathbb{G}_1 \times \mathbb{G}_2$. In our scheme, we do not use the public key of $\tilde{H}$, so it is not necessary to calculate it. (For details see chapter 4) in the extended version.
   - Sample a random value $r \leftarrow \mathbb{F}_q$.
   - Run **PRF.KeyGen**$(\lambda) \to (K, pp)$ to build $F_K : \{0,1\}^* \to \mathbb{G}_1 \times \mathbb{G}_2$. In this adaptation, we need F_K to be computationally indistinguishable from a function that outputs $(R, S) \in \mathbb{G}_1 \times \mathbb{G}_2$ such that $Dlog_g(R) = Dlog_h(S)$ is uniform over $\mathbb{F}_q$ (i.e. $e(R, h) = e(g, S)$).
   - Run $CFEval_\tau^{off}(K, f) \to w_f$, called concise information for f. (For details on PRF and CFEval see Section 5.4.3).
   - Set $SK = (pk, sk, \kappa, r, K, w_f)$ and $PK = (pk, pp, f)$.
2. **ProbGen**$_{SK}(\overrightarrow{x} = (x_1, \ldots, x_n)) \to \sigma_x, \tau_x$, requiring the operations below:
   - Choose an arbitrary string $\Delta \in \{0,1\}^\lambda$ (identifier for $\overrightarrow{x}$).
   - For i=1 to n:

(a) Run BFV.Enc(x_i)=$\mu_i \in R_q^2$ and compute its hash value $(T_i, U_i) = \tilde{H}_\kappa(\mu) \in \mathbb{G}_1 \times \mathbb{G}_2$. Next run $\mathrm{F}_K(\Delta, i) = (R_i, S_i) \in \mathbb{G}_1 \times \mathbb{G}_2$.

(b) Compute $X_i = (R_i \cdot T_i^{-1})^{1/r}$ and $Y_i = (S_i \cdot U_i^{-1})^{1/r} \in \mathbb{G}_1, \mathbb{G}_2$ respectively.

(c) Set $\sigma_i = (T_i, U_i, X_i, Y_i, \Lambda_i = \mathbb{1}_{\mathbb{G}_\mathbb{T}}) \in (\mathbb{G}_1 \times \mathbb{G}_2)^2 \times \mathbb{G}_T$. We denote the level of tag as $lev(\sigma_i)$ and we set $lev(\sigma_i) = 1$.

- Set $\sigma_x = (\Delta, \mu_1, \sigma_1, \ldots, \mu_n, \sigma_n)$ and $\tau_x = \perp$.

3. **Compute**$_{PK}(\sigma_x) \to \sigma_y$ consisting of the following steps:

- Let f be an admissible circuit.

- Run the evaluation circuit f over the **BFV** encrypted data and obtain $\mu =$ **BFV.Eval**$(f, \mu_1, \ldots, \mu_n)$. Let us note that, for preserving the homomorphic properties of the hash, the difference with the normal evaluation of the BFV scheme is that here the multiplication of polynomials is performed over R_q without the *mod* $\Phi_m(x)$-reduction and without the rounding step, and we assume thus that this modulus reduction and this rounding operations are performed at the end by the verifier receiving the result of the evaluation of f.

- Apply (gate-by-gate) f over the authentication tags $(\sigma_1, \ldots, \sigma_n)$, using the following gate functions **GateEval()**. **GateEval**:$(f_g, \sigma_1, \sigma_2) \to \sigma$
Parse $(T_i, U_i, X_i, Y_i, \Lambda_i) \in (\mathbb{G}_1 \times \mathbb{G}_2)^2 \times \mathbb{G}_T$ for $i = 1, 2$, where f_g stands for tag addition ("+") or tag multiplication ("×"). We try to compute $\sigma = (T, U, X, Y, \Lambda)$.

  - **Add two tags together.** If $f_g =$"+", the addition takes different forms depending on the levels of the input tags.
  If $lev(\sigma_1) = lev(\sigma_2)$, then $\sigma = (T_1 \cdot T_2, U_1 \cdot U_2, X_1 \cdot X_2, Y_1 \cdot Y_2, \Lambda_1 \cdot \Lambda_2)$ with $lev(\sigma) = 1$.
  Else, without loss of generality, let suppose that $lev(\sigma_1) = 1$ and $lev(\sigma_2) = 2$(i.e. there is a multiplication gate before this gate).
  The idea is to create a level-2 tag (σ_1') from σ_1 as follows: $\sigma_1' = (e(T_1, h), e(g, U_1), e(X_1, h), e(g, Y_1), \Lambda_1)$. Then compute $\sigma = \sigma_1' + \sigma_2$ as in the first case but set $lev(\sigma) = 2$.

  - **Add a constant to a tag** $(c + \sigma_1)$**.** This method depends on the level of the tag as follows:
  If $lev(\sigma_1) = 1$, then the result tag $\sigma = (T_1 \cdot (g^c), U_1 \cdot (h^c), X_1, Y_1, , \Lambda_1)$.
  If $lev(\sigma_1) = 2$, then we obtain: $\sigma = (T_1 \cdot (e(g, h)^c), U_1 \cdot (e(g, h)^c), X_1, Y_1, Z_1, \Lambda_1)$.
  In both cases $lev(\sigma) = lev(\sigma_1)$.

  - **Multiplication by a constant** $(c \cdot \sigma_1)$**.** The result tag is $\sigma = (T_1^c, U_1^c, X_1^c, Y_1^c, \Lambda_1^c)$ and $lev(\sigma) = lev(\sigma_1)$.

  - **Multiplication**. For $f_g =$"×" on two tags $(\sigma_1 \times \sigma_2)$
  If $lev_y(\sigma_1) > 1$ or $lev_y(\sigma_2) > 1$ then reject. **Else** calculate $T = e(T_1, U_2)$, $U = e(T_2, U_1)$, $X = e(X_1, U_2) \cdot e(X_2, U_1)$, $Y = e(T_2, Y_1) \cdot e(T_1, Y_2)$, $\Lambda = e(X_1, Y_2)$. Also set $lev(\sigma) = 2$.
  It is not necessary to keep U and Y after a multiplication because $T = U$ and $X = Y$. We keep them only for the sake of clarity. As noted in [3],

one can see the function f as the composition of two functions $f_g(f_1, f_2)$ in the last gate f_g of f.

– Set $\sigma_y = (\Delta, \mu, \sigma)$, where σ is the tag obtained after evaluating the last tag of f.

4. $\mathbf{Verify}_{SK}(\sigma_y, \tau_x) \rightarrow (acc, y)$, for $\sigma_y = (\Delta, \mu, \sigma)$, using the following operations:

- Compute $\tilde{\mathbf{H}}_\kappa(\mu) \rightarrow \tilde{\nu}$.
- Run $\mathbf{CFEval}^{on}_{\Delta}(K, w_f) \rightarrow W$(see Section 5.4.3 for details on the online closed-form method).
- Check, depending on the of degree of f, as follows:

  (a) If deg(f)=1, check the following equations:

$$(T, U) = \tilde{\nu}(= (g^{((\mu(\alpha))(\beta))}, h^{((\mu(\alpha))(\beta))}))$$
$$e(X, h) = e(g, Y)$$
$$W = e(T \cdot X^a, h).$$

  (b) Else, check over $\mathbb{G}_T$ the following equations:

$$T = U = \tilde{\nu}(\text{ i.e. } = e(g, h)^{((\mu(\alpha))(\beta))}) \tag{5.1}$$
$$X = Y \tag{5.2}$$
$$W = T \cdot (X)^r \cdot (\Lambda)^2 \tag{5.3}$$

- If all equations are satisfied set the check bit **acc** to 1 (accept), otherwise set it to 0 (reject).
- Finally, if **acc**=1, $\mu' = \mu \ mod \ \Phi_m(x) = (c_0, c_1, c_2)$ and set $\mu' = (\lceil t \cdot c_0/q \rfloor, \lceil t \cdot c_1/q \rfloor, \lceil t \cdot c_2/q \rfloor)$ $y = BFV.Dec_{dk}(\mu')$, otherwise set $y = \perp$.

Theorem 6. If **BFV** is a semantically secure homomorphic encryption scheme, $\tilde{\mathbf{H}}$ is a collision-resistant homomorphic hash function and **F** is a pseudorandom function, then $\mathcal{VC}$ described above is correct, adaptive secure and input private.

Proof. Same proof as for the scheme $\mathcal{VC}_{quad}$ from [3]. ■

5.6 VC and FHE for first layer

In this section, we present more in details our architecture for partially encrypted machine learning using Verifiable Computing for BFV homomorphic encrypted data.

As illustrated in Figure 5.2, the client sends the homomorphic data encrypted at the server along with a authentication tag. The server computes the first layer (f) of a neural network on the homomorphic encrypted data, generates a short proof-calculation for verifying the homomorphic results and sends them to an operator. He later on checks using the short proof that the calculation of the first layer is correct and, if so, he decrypts the result of this first layer and completes the neural network on clear data. More precisely, the user runs the **ProbGen** algorithm (described below) to encrypt and to generate

a tag corresponding to his/her data. We note that a preliminary step consists in the generation of the keys by the operator (**Setup** algorithm). The server runs the **Compute** function (described below) over the received data to apply f, the first layer of the neural network and to compute the tag associated with the result. It returns thus the ciphertext $Enc(f(x))_{BFV}$ and the result tag $\sigma = f(\sigma_i)$ to the operator which verifies the results it receives with the **Verify** function. If the calculation is correct, he decrypts the result using the homomorphic secret key and he completes the evaluation of the remaining of the neural network over the clear data for obtaining the prediction result.

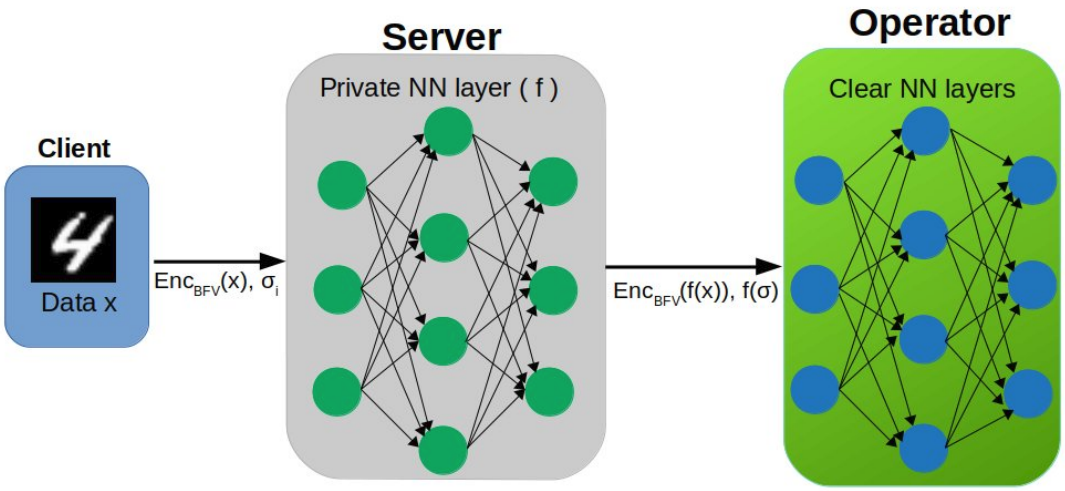

Figure 5.2: Semi-encrypted neural network using FHE and VC.

Let us now go into more details. The data represented as $x = (x_0, \ldots, x_n)$ is encrypted with a BFV cryptosystem. For authentication the client uses the secret key to generate a series of tags $(\sigma_1, \ldots, \sigma_n)$, that will help the server produce (without any secret key) the authentication tag σ corresponding to the result of the first private layer of the neural network, i.e. the quadratic activation function f (**Compute** algorithm). This tag $\sigma = f(\sigma_1, \ldots, \sigma_n)$ authenticates the ciphertext $\mu = f(\mu_0, \ldots, \mu_n)$ using the properties of homomorphic BFV ciphertexts obtained. The one receiving $f(\mu_0, \ldots, \mu_n)$ can verify effectively that the server performed the computation correctly (using the secret key of $\mathcal{VC}$) and can decrypt it to obtain $f(m_0, \ldots, m_n)$ (using the homomorphic secret key). This decrypted result is the input of the remaining of the neural network performed on clear data (Clear-NN algorithm).

Our steps are specified as follows:

Setup(NN, λ): Takes as input the neural network and generates the public (PK) and secret key (SK) to the VC scheme for BFV data.

ProbGen$_{PK}(\overrightarrow{x} = (x_1, \ldots, x_n))$: Takes as input the data $\overrightarrow{x}$. For all $i \in [1, n]$, it generates in parallel the encrypted $\mu_i \leftarrow BFV.Enc_{pk}(x_i)$ and the tags σ_i corresponding to the μ_i as shown in the above section.

Finally, it outputs $\sigma_x = (\mu_1, \sigma_1, \ldots, \mu_n, \sigma_n)$ and $\tau_x = \perp$.

Compute$_{PK}(\sigma_x)$: Taking as input the encrypted data and the corresponding tags, it runs the evaluation circuit BFV f over the BFV encrypted data μ_i, and, in the same time, it generates the tag corresponding to the evaluation of the circuit f over the tags

σ_i gate-by-gate as mentioned above in the GateEval algorithm. Finally, it returns $\sigma_y = (\Delta, \mu, \sigma)$.

Complete$_{SK}(\sigma_y, \mu)$: Taking as inputs the tag and the encrypted result, it verifies the calculation using $\mathcal{VC}.Verify_{SK}(\sigma_y, \tau_x)$ and if it is true, it decrypts the result and completes the remaining of neural network Clear-NN over $f(x_1, \ldots, x_n)$, else it refuses the result.

The security of this architecture such as defined in section 5.3 is based on the security of VC over BFV encrypted data and under the hypothesis of non collusion between these three entities.

In this architecture, we can evaluate an activation function $f : \mathbb{F}_q^n \to \mathbb{F}_q$ of degree at most 2, because our adaptation of $\mathcal{VC}$ works for a multi-variate function of degree at most 2. We can also hide the function f from the server, using the same modification proposed in [3] in the two algorithms **KeyGen** and **Compute** (namely, by modifying the multiplication-by-constant method, using $\tilde{H}_K(Enc_{BFV}(c))$ instead of c, which requires the modification in the algorithms cited above).

5.7 Experimental Results

We present here the experimental results of applying our approach for the digit recognition on the standard MNIST dataset.

In this section we work more to characterize the computational performances of our architecture than really building an operational machine learning system. In other words, despite that we use a small dataset size, this allows us to obtain a representative view for our architecture in terms of execution times and performances.

Hardware and Software. Let us precise that all tests were performed on an 2016 DELL PC(Genuine-Intel Core $i7-6600U$, 4 cores at 2.60GHz with 16GB RAM at 2.13GHz), on Ubuntu (linux kernel 4.15.0-91-generic, with the architecture $x86-64$) as operating system.

Choosing a model. For the training, we apply the adversarial training approach from [153]. They learned $P \in \mathbb{Z}^{d \times n}$ and $(D_i)_{i \in [\ell]} \in (\mathbb{Z}^{d \times d})^\ell$, with the model defined as $f_i(x) = (Px)^T D_i(Px)$, $\forall i \in [\ell]$. Then, they generalized this model by adding a bias term: $f_i(x) = (Px + b)^T D_i(Px + b)$ for $b \in \mathbb{Z}_p^d$, and, for simplicity, they used an equivalent of this model by systematically adding a 1 at the beginning of x when encrypting it $x' = (1, x_1, \ldots, x_n)^T$. The prediction for the class of $x \in [0, 255]^{785}$ is $argmax_i(f_i(x))$ for $i \in [\ell]$. This modelling is important for FE efficiency [153], because it reduces the number of pairing computations. In our implementation, we used an equivalent model g defined as $g(x) = Q^t(Px)^2$, where $Q \in \mathbb{Z}^{d \times \ell}$ and $Q[i, j] = D_j[i, i]$ (i.e. $f_i(x) = g_i(x) = Q_i^T(Px)^2$ with Q_i the i-th row of Q). The prediction for the class of $x \in [0, 255]^{785}$ is $argmax(g(x))$. As such, instead of using a matrix per label, we use a new matrix Q for all labels. Therefore, the resulting model is a polynomial network of degree 2 with one hidden layer of d neurons and a square for the activation function.

Implementation tools.

Homomorphic Encryption. We use the SEAL library [154], a homomorphic encryption library developed by Microsoft and written in modern standard C++. In terms of security,

we choose parameters for providing 128 bits of security. We run SEAL with the following parameters: $n' = 4096$, $log_2(q) = 109$ and $t = 1032193$. These parameters are chosen using the Homomorphic Encryption Standardization report [155].

The table 5.2 illustrates the evolution of the noise budget for the prediction, and, as expected, the noise growth caused by the homomorphic multiplications increases rapidly (in our case $h_i \times h_i$ grows the noise by 38 bits).

	$[x_i]_{BFV}$	$h_i = P_i \cdot [x]_{BFV}$	h_i^2	$Q_i^2 \cdot h$
Noise budget	45 *bits*	40 *bits*	8 *bits*	5*bits*

Table 5.2: Noise budget where Q_i and P_i are the $i-th$ row of Q and $i-th$ row of P respectively.

Verifiable computing. We use the HAL library [156], a library for Homomorphic Authentication over encrypted BGV data, written in C and providing 128 bits of security, by using the Barreto-Naehrig curve for pairings.

In our experiments, we encrypt and decrypt homomorphically the data with SEAL library and we use the HAL library for authentication but for BFV encrypted data.

5.7.1 Results

Our tests consist in classifying a MNIST image data, a greyscale RGB image with 784 pixels, represented as a vector $x \in [0, 255]^{784}$. As illustrated in Figure 5.3, we add 1 at the beginning of x when encrypting it (encrypting pixel by pixel) by the user. The server evaluates the model g over encrypted data (Hidden layer). Now the operator runs the Clear-NN algorithm for verifying the results and decrypting it to obtain $g(x)$ and calculate the argmax(g(x)). Our model achieves $97,54\%$ accuracy on a test set of 10000 labeled images. We note that in our test we obtain the same confusion matrix as for the FE-model (see Figure 4 in [153]).

Performance. In Table 5.3, we describe the time evaluation for our approach. We remark that the user can execute the encryption function and the tags generation in parallel, so the user runs this step in average in less than 2.5s. (Let us note that this time is inferior to the time of user for encryption using the $FE-$mode, of 8s.) Similarly, the server calculates the quadratic function g over encryption and over the authentication tag in parallel. Then, the server execution time is less than 3.8s. We note that for computing the function g for all labels, we run in parallel the $g_i(\sigma_1, \ldots, \sigma_{785})$ for $i \in [10]$. Finally using our architecture, the time on the operator side is negligible. Namely, the operator time is 0.021s (decryption and verification together), while the time for $argmax$ on the decrypted results is negligible, as expected. In terms of memory requirements, Table 5.4 describes the size in KiloBytes of the data used our architecture. More precisely, we report the size of the homomorphic ciphertexts and of the authentication tags on both user and server sides.

5.8 Conclusion

In this work, we presented a solution for private classification of sensitive data based on Homomorphic Encryption combined with a Verifiable Computing (VC) protocol to en-

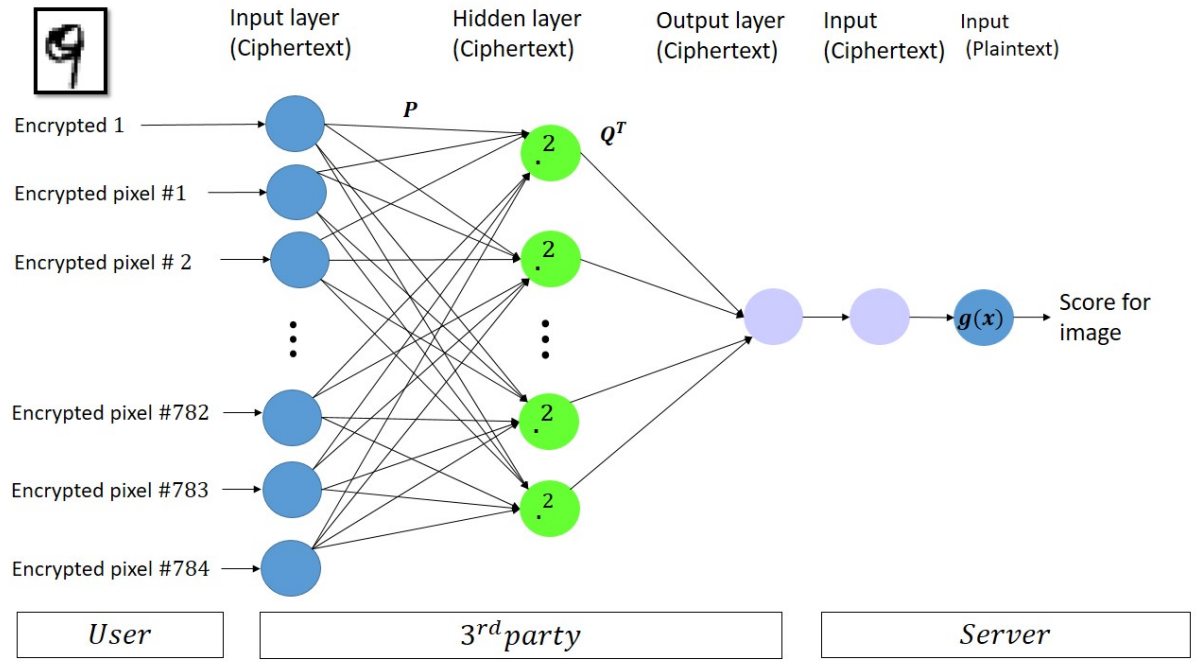

Figure 5.3: Overview of our architecture with model g.

	User-side		***Server-side***		***Operator-side***	
Operation	*Enc*	*GenTag*	$g(Enc(x))$	$g(\sigma_1, \ldots, \sigma_n)$	*Verify*	*Dec*
time	1.760	2.525	3.8	3.35	0.015	0.006

Table 5.3: Costs (in seconds) for our architecture, where $x = (x_1, \ldots, x_{785})$

	User-side		***Server-side***	
	$Enc(x_i)$	$tag(\sigma_i)$	$Enc(f(x))$	$f(\sigma_i)$
Size	194	0.408	291	1.2

Table 5.4: Size (in KB) for MNIST test, where $x = (x_1, \ldots, x_i, \ldots, x_n)$, with $i = \{1, \ldots, 785\}$.

sure the result integrity. We built on a semi-encrypted neural-network trained using a semi-adversarial model [4] and then preserve the confidentiality of sensitive data and the integrity for treatments using an application of VC over BFV encrypted data. Our experimental results for the MNIST image dataset are encouraging giving good classification accuracy (nearly 97.54%) with decent execution performances (less than 6s for the overall protocol). However, due to the limitations on the classes of functions supported in today practical VC techniques for encrypted data, our work was to some extend restricted to a private evaluation only for a first quadratic layer and we had to finalize (on another entity) the rest of the classification process on clear data from the decrypted intermediate values.

As such, one open research problem worth investigating consists in developing efficient verifiable delegation protocols with support for the computation of a broader class of functions, in particular any multi-variate polynomials. This will allow us to provide more complete privacy and integrity solutions for the evaluation of neural networks. Another more concrete research line we plan to follow is to improve the performances of the

proposed approach by exploring the use of batching and other optimization techniques dedicated to HE computation.

Chapter 6

Secure FL using VC and HE

6.1 Introduction

In recent years, machine learning solutions and in particular deep learning ones are widely employed in different domains such as healthcare, autonomous driving, finance and computer vision. However, this raises several security and privacy issues since the learning step requires access to massive amounts of heterogeneous data, part of which may be sensitive or private information.

Federated Learning (FL), an emerging recent training setting, allows to collaboratively train a model under the coordination of a central server or service provider without data outsourcing. As such, the data of each client remain stored locally without being shared and only the successive models are disclosed [157]. This is interesting for various reasons such as coping with the sensitive nature of training data, privacy laws and data regulations

(*e.g.* GDPR [158], HIPAA [159], etc.) or business requirements. Even if this paradigm offers privacy improvements over a traditional centralized training model, recent research shows that attackers can indirectly retrieve private client data (based on the shared model updates, see [160], [161], [162] for more details). Moreover, there are cases in which the model itself is sensitive (*e.g.* due to proprietary reasons) or subject to attacks from/on the coordination server in order to alter it or modify the resulting inference capabilities.

In this context, there have been recent proposals to improve data privacy and model confidentiality through emerging cryptographic techniques such as Homomorphic Encryption [163, 164] or Multi-Party Computation [165, 166]. However, none of these approaches address integrity issues with respect to the computations performed to build the global model or to the clients' behavior. To the best of our knowledge, VerifyNet [167] is the only work proposing a privacy-preserving training for "cross-device" FL (the clients are a very large number of mobile or IoT devices) with guarantees of integrity of the global model, using a double-masking protocol and a homomorphic hash function.

We complete the picture of this contribution by proposing a novel secure approach for Federated Learning guaranteeing privacy protection for the training data and integrity for the overall model, using Homomorphic Encryption and Verifiable Computation. Our solution, which addresses threats coming from the server, is intended mainly for the "cross-silo" FL setting [168] in which the training involves only a small number of clients (that we suppose reliable), such as several organizations (e.g. medical, financial, insurance) collaborating to train a model.

In this chapter, we make the following main contributions:

- We present a secure framework for privacy-preserving and verifiable FL relying on Homomorphic Encryption and Verifiable Computation. Besides the general architecture, we also give some examples of practical use cases for our secure federated learning solution.

- We concretely instantiate our framework using the Paillier additive homomorphic scheme associated to the LePCoV primitive [129] which is a linearly homomorphic Authenticated Encryption with Public Verifiability scheme derived from [130]. It allows us to authenticate the computation of the global training model over the homomorphically encrypted data, with the additional guarantee that the correctness of the result can be publicly verified.

- In order to accelerate the performances of the system, we also propose a batching approach for the homomorphically encrypted data. We then give extensive performances results on the FEMNIST dataset and we study the accuracy of the resulting models and the overhead induced by using our homomorphic encryption and authentication primitives. All these results have been obtained using an efficient C/C++ prototype implementation of the framework written as part of this work.

- We provide a security and threat analysis for our FL approach. As such, under the security hypothesis we made, we ensure that both the clients local data and the updates to the gradients remain undisclosed to the server and that a malicious server cannot tamper or misuse the model while training.

6.2 Related work

6.2.1 Secure Federated Learning and Homomorphic Encryption

Most of the work consisting in applying homomorphic encryption to machine learning models concentrate on making the inference on private encrypted data (*e.g.* CryptoNets [132], TAPAS [141], NED [142]) and not so much on the training.

The first works addressing the problem of privacy-preserving machine learning training concentrated on a centralized setting where all the data are outsourced and the models are only linear [169, 170]. As for the few approaches proposing a complete centralized training of neural networks on homomorphic encrypted data, they have quite impractical performances or huge cryptographic parameters ([171]).

Other works propose solutions in the case of multi-servers either for clustering or regression. A lot of recent approaches employing homomorphic encryption are proposed for a collaborative distributed learning in which there is no central server mostly for linear models [172, 173] and, more recently, for neural networks [174].

As for the case of cross-silo FL, there are only a few recent papers, proposing the use of homomorphic encryption (usually additive) to ensure the secure computation of the global model ([163],[164], [175]). The first two approaches are only theoretical and the third one uses different datasets for the validation. Morever, all the above approaches are under the classical hypothesis of an honest but curious central server, without making use of any verifiable computing protocols to ensure the global model integrity.

6.2.2 Secure Federated Learning and Verifiable computation

In order to solve the problem of privacy protection and verifiability in deep learning system, several works have been proposed, like SafetyNets [146], and Slalom [176]. However, these schemes either support a small variety of activation functions like in [146] or require additional hardware assistance as in [176].

As for applying the verifiable computing protocols for the FL setting (i.e. verify the integrity of the aggregated results returned from the central server), to the best of our knowledge, there is only a recent work on this subject. Xu et al. [167] introduce the VerifyNet architecture as a solution for cross-device FL preserving the integrity and the confidentiality of the model with regards to the global server. The verification of the server calculation results (i.e. the aggregated model) is realized using the homomorphic hashes and pseudorandom functions. The privacy of user's gradients is guaranteed via a double-masking protocol, having the inconvenient that it requires multiple exchanges between the users.

As seen in this section and, to the best of our knowledge, so far there are no approaches for a secure FL which supports the integrity of the server calculation results while guaranteeing the model privacy by means of verifiable computation and homomorphic encryption.

6.2.3 Secure Federated Learning and other Multi-Party Computation

Multi-party computation (MPC) protocols allow several parties to collaborate in order to compute a function on their private data such that each party knows only its input and output. There are several FL approaches using MPC [165], [166] but due to the high-communication costs of the multi-party computation and the inherent distributed nature of FL, it is difficult to implement efficient methods.

6.2.4 Secure Federated Learning and Differential Privacy

A few works [177–180] have implemented differential privacy to protect clients' data from other clients or end-users in a FL context. These works suggest that differential privacy is more appropriate for cross-device FL applications. Indeed, a high number of clients is required to simultaneously allow that

- the ratio of participants per round is low, thus limiting the probability that a given client participates and therefore (indirectly) releases any information about his training data in the considered round
- the absolute number of participants per round is high, reducing the sensitivity of the model updates in a client-level differential privacy point of view

This hypothesis is not reasonable in the context of this contribution where the number of clients does not exceed a few hundreds. That is why we focus in this chapter on scenarios in which we consider that the recipients of the final model (clients and end-users) do not perform attacks like membership inference [181, 182] or model inversion [160, 162] on the model updates or the final model. Properly implementing differential privacy would need further experiments and we wish to design such a framework in a future work.

6.3 Preliminaries

6.3.1 Federated Learning (FL)

FL is a decentralized framework that enables multiple clients to collaboratively train a shared global model under the orchestration of a central server while keeping the training data distributed on the client devices. As a starting point, the server initializes a global model randomly and then the FL process consists of multiple rounds. At the beginning of each round, the server selects a subset of clients that take part in training and sends to them the current global model. Next, each selected client trains the model locally on his own data and communicates only the model updates back to the server. Finally, the server aggregates these updates before accumulating them into the global model thereby concluding the round. The most common approach to optimization for FL is the Federated Averaging algorithm [157]. Here, each client runs several epochs of minibatch stochastic gradient descent (SGD) minimizing a local loss function, and then the central server performs a weighted averaging of the updated local models to form the updated global model. Pseudocode is given in Algorithm 1. By removing the need to aggregate all data on a central server, FL helps to ensure data privacy and reduces communication costs. As a result, FL applies best in situations where data are privacy-sensitive or large in such a way that it is undesirable or infeasible to transmit them to the server.

Algorithm 1 Federated Averaging M is the total number of clients; K is the number of participants per round t; the selected clients are indexed by k with $\mathbf{D_k}$ the training set of data points on client k and $n_k = |\mathbf{D_k}|$; B is the local minibatch size; E is the number of local epochs; μ is the learning rate; w are model parameters and l is the local loss function.

Server executes:

initialize w_0
for each round t **do**
  $K_t \leftarrow$ random set of $K \leq M$ clients
  for each client $k \in K_t$ in parallel **do**
    $(w_{t+1}^k, n_k) \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$ where $n = \sum_{k=1}^{K} n_k$

ClientUpdate(k, w)**:**

initialize $w_k = w$
$\mathcal{B} \leftarrow$ split n_k samples of $\mathbf{D_k}$ into batches of size B
for each round epoch from 1 to E **do**
  for each batch $b \in \mathcal{B}$ **do**
    $w_k \leftarrow w_k - \mu \bigtriangledown l(w_k; b)$
return (w_k, n_k)

6.3.2 Homomorphic Encryption (HE)

Homomorphic Encryption (HE) schemes allow to perform computations directly over encrypted data without decrypting it first. That is, with a Fully homomorphic Encryption scheme E, we can compute $\mathsf{E}(\mathsf{m_1} + \mathsf{m_2})$ and $\mathsf{E}(\mathsf{m_1} \times \mathsf{m_2})$ from Encrypted messages $\mathsf{E}(\mathsf{m_1})$ and $\mathsf{E}(\mathsf{m_2})$. Thus homomorphic encryption provides a way to outsource computations to the cloud while protecting the data confidentiality. Moreover, a simple additive homomorphic cryptosystem (*i.e.* allowing to obtain only the encryption of the addition of two messages) is enough to perform secure federated averaging. Let us recall the general principles of the **Paillier cryptosystem**, a well-known and popular additive homomorphic scheme [22].

KeyGen$(sz) \rightarrow (pk, sk)$: It generates the keys for the cryptosystem taking as input the number of bits sz of the modulus.

Choose two large prime numbers p_E and q_E such that $\lambda = \text{lcm}(p_E - 1, q_E - 1)$, and set $N_E = p_E q_E$. We note that the cleartext domain is $\mathbb{Z}_{N_E}$ and the ciphertext domain is $\mathbb{Z}_{N_E^2}$.

Select a random $g < N_E^2$ such that $\gcd(L(g^\lambda \mod N_E^2), N_E) = 1$,with $L(u) = \frac{u-1}{N_E}$.

Set $pk = (N_E, g)$ and $sk = (p_E, q_E)$.

Enc$_{pk}(m) \rightarrow c$: It produces a ciphertext c using the public key pk by computing $c = g^m r^{N_E} \mod N_E^2$, where $m < N_E$ is the message and r is uniformly chosen in $\mathbb{Z}_{N_E}$.

Dec$_{sk}(c) \rightarrow m$: It computes the plaintext m from the ciphertext c, using the private key sk.

Letting $D = L(g^\lambda \mod N_E^2)$ and D^{-1} its multiplicative inverse in $\mathbb{Z}_{N_E}$, the decryption is

performed by evaluating

$$m = \text{Dec}(c) = L(c^{\lambda} \mod N_E^2) \times D^{-1} \mod N_E.$$

More importantly, for the present purpose, this cryptosystem has the following homomorphic properties:

1. $\text{Dec}(\text{Enc}(m_1)\text{Enc}(m_2)) \mod N_E^2 = m_1 + m_2 \mod N_E$ (addition of two encrypted messages).
2. $\text{Dec}(\text{Enc}(m)g^k) \mod N_E^2 = m + k \mod N_E$, for all $k \in \mathbb{Z}_{N_E}$ (addition of an encrypted message to a clear integer).
3. $\text{Dec}(\text{Enc}(m)^k) \mod N_E^2 = km \mod N_E$, for all $k \in \mathbb{Z}_{N_E}$ (multiplication of an encrypted message by a clear integer).

6.3.3 Batching for Paillier

We can batch several plaintext messages m_i in a same Paillier ciphertext, each one represented on t bits. Let b be the size of a batch, i.e. the maximum number of positive messages we can encrypt in a same Paillier ciphertext $c_p = g^{m_1+m_2+\ldots+m_b} * r^{N_E} \mod N_E^2$ or written differently $Enc(m_1|m_2|\ldots|m_b)$. To decrypt correctly c_p it follows that $|m_1 + m_2 + \ldots m_b| \leq N_E$ and $b \leq log(N_E)/t$ (i.e. with a modulus of 2048 bits and messages of $t = 64$ bits, it is possible to pack together max 32 messages).

However, if we want to perform addition on these packed ciphertexts, one must take this into account to set up the dimension of the initial batch to avoid an overflow. Let nb_{add} the number of additions we want to perform on these packed messages. The padding (i.e. the number of zero bits) that has to be added to each slot is equal to nb_{add}.

As such, one has to encrypt $Enc(m_1\ 0\ldots0|\ m_2\ 0\ldots0|\ldots|\ m_b\ 0\ldots0)$. The size of the batch will then have to be at most: $b = \left\lfloor \frac{log_2(N_E)}{t+nb_{add}} \right\rfloor$.

Let us give a short example. Suppose each message we want to encrypt is a real positive number between 0 and U, with $U = 100$ and m_i having 4 representative digits after the decimal point. As such to represent these messages, one must have at least $\lceil log_2(U * 10^4) \rceil$, i.e. 20 bits. To perform two additions on ciphertexts batching these type of messages, for a modulus of 2048 bits, one can pack $2048/(20 + 2) = 93$ messages in a single ciphertext. For 100 additions on the ciphertexts of the same format, one can have at most 17 slots/ciphertext.

6.3.4 Verifiable Computation

Verifiable Computation (or Verifiable Computing) VC is a cryptography tool meant to secure the integrity of computations on authenticated data. It enables a client to delegate to another entity (in most cases a server) the computation of a function. The other entity evaluates the function and returns the result with a proof that the computation of the function was carried out correctly.

Now, we present the VC for Paillier cryptosystem [129]. It is a Linearly Homomorphic Authenticated Encryption scheme with Public Verifiability (LAEPuV) and provable correctness called LEPCoV and it allows the public verifiability of data returned by the

server. This scheme improves Catalano et al.'s instantiated scheme [130] by avoiding false negatives during the verification step.

Let $S = (KeyGenS, Sign, Verify)$ be a signature scheme.

- $AkeyGen(sz, I)$: takes as input a prime size sz (in number of bits) and an integer I representing the upper bound for the number of messages encrypted in each dataset. It calculates the secure sk and public pk parameters as follows: sample four (safe) primes p_E, q_E, p_S, q_S of size $sz/2$, such that $N_E = p_E \cdot q_E$ and $N_S = p_S \cdot q_S$ it holds that $\varphi(N_S) = (p_S - 1)(q_S - 1)$, the group elements g_0, g_1, h_1,..., $h_I \in \mathbb{Z}^*_{N_s}$ and $g \in \mathbb{Z}^*_{N_E^2}$ of order N_E, and picks a hash function H. Finally, it runs $KeyGenS(sz)$ to obtain the secure and private signature key (sk_S, pk_S). Returns the key pair (sk, pk), where $pk = (N_E, g, N_S, g_0, g_1, h_1, \ldots, h_I, H, pk_S)$ and $sk = (p_E, q_E, p_S, q_S, sk_S)$.

- $AEncrypt(sk, \tau, i, m)$: probabilistic algorithm that takes as input the secret parameter, a message $m \in M$, a dataset identifier τ, and an index $i \in \{1, \ldots, I\}$ to calculate the ciphertext c containing the encryption of the message with the tag of verification. Thus, it computes the Paillier encryption C of the message m, $R = H(\tau||i)$, and a tuple $(a, b) \in \mathbb{Z}_{N_E} \times \mathbb{Z}^*_{N_E}$such that $g^a b^{N_E} = CR \mod N_E^2$ (using the factorisation of N_E). In addition, if τ is used for the first time, it chooses a not yet used prime e of length $l \leq sz/2$ such that $gcd(eN_E, \varphi(N_S)) = 1$, it computes its inverse e^{-1} mod $\varphi(N_S)$, and its signature $\mu_e = Sign_{sk_S}(\tau||e)$ and it stores (τ, e, e^{-1}, μ_e) in the list L. Otherwise, it takes (τ, e, e^{-1}, μ_e) from the list L. Then, it chooses an element s uniformly at random from $\mathbb{Z}_{eN_E}$ and it computes x using the p_S and q_S such that $x^{eN_E} = g_0^s h_i g_1^a \mod N_S$. It returns $c = (C, e_i, e_i^{-1}, \mu_e, \tau, \sigma)$, where $\sigma = (a, b, s, x)$ is the verification tag.

- $AEval(pk, \tau, f, \{c_i\}_{i\in[I]})$: takes as input the public key pk, a dataset identifier τ, a linear function $f = (f_i)_{i\in I}$ and I ciphers $\{c_i\}_{i\in[I]} = (C_i, e_i, e_i^{-1}, \tau_i, \sigma_i)$. The output is a cipher c. The algorithm checks if there exists an index $l \in [I]$ such that $\tau \neq \tau_l$, or that the signature $(Verify(pk_S, \tau||e_l, \mu_{e_l}) = 0)$. Furthermore, the algorithm checks if there are two indexes $i \neq j \in [I]$ such that $e_i \neq e_j$. If one of the checks is true, the algorithm aborts. Otherwise, the algorithm sets $e = e_1, e^{-1} = e_1^{-1}$, $\mu_e = \mu_{e_1}$ and evaluates f over ciphertext as: $C = \prod_{i=1}^{I} C_i^{f_i} \mod N_E^2$. It also evaluates f over the tag to obtain a new tag (a, b, s, x) as follows: $a = \sum_{i=1}^{I} f_i a_i \mod N_E$, $b = \prod_{i=1}^{I} b_i^{f_i} \mod N_E^2$, $s = \sum_{i=1}^{I} f_i s_i \mod eN_E$, $s' = \left(\sum_{i=1}^{I} f_i s_i - s\right)/(eN_E)$, $a' = \left(\sum_{i=1}^{I} f_i a_i - a\right)/N_E$, and $x = \frac{\prod_{i=1}^{I} x_i^{f_i}}{g_0^{s'} g_1^{a' e^{-1}}} \mod N_S$.
  It returns the cipher $c = (C, e, e^{-1}, \mu_e, \sigma)$.

- $AVerify(pk, \tau, c, f)$: takes as input the public key pk, a dataset identifier τ, a cipher $c = (C, a, b, e, s, \tau, x)$, and a linear function $f = (f_i)_{i\in[I]}$, to detect if c is a valid or invalid cipher. For this goal, the algorithm checks that:

$$\begin{cases} Verify(pk_S, \tau||e, \sigma_e) = 1; \\ a, s \in \mathbb{Z}_{eN_E}; \\ x^{eN_E} = g_0^s \prod_{i=1}^{I} h_i^{f_i} g_1^a \mod N_S; \\ g^a b^{N_E} = C \prod_{i=1}^{I} H(\tau||i)^{f_i} \mod N_E^2; \end{cases}$$

  If all checks pass, it outputs 1 (i.e c is a valid cipher), else it outputs 0, (i.e. c is an

invalid cipher).

- $ADecrypt(sk, \tau, c, f)$: Taking as input the secret parameter sk, a data set identifier, a cipher c, and a linear function $f = (f_i)_{i\in[I]}$, it calculates the decryption of c or $\perp$ (if c is invalid cipher). Then it verifies if c is a valid cipher by running $AVerify$ (pk, τ, c, f). If passed, the algorithm returns the message m obtained by $Dec(c)$. Otherwise, it returns $\perp$.

For lack of space, we refer the reader to [130] for the correctness and the security proofs for LAEPuV scheme and to [129] for the security and correctness proofs of the LEPCoV scheme.

6.4 A secure framework for Confidential and Verifiable Federated Learning

6.4.1 Overview of the architecture

Figure 6.1 shows the high-level view of the FL framework while training with a total of M clients in the case of a malicious central server. In our approach, the confidentiality of the model is ensured by homomorphic encryption and the integrity for the computation of the global model by the central server is guaranteed by public verifiability.

Before the training actually begins, the central server shares with the M clients the global architecture of the neural network that will be trained. Also, once the enrollment of the clients participating to the training is completed, the keys necessary for the homomorphic encryption and the signatures are generated by one of the clients. This client responsible for the key generation can be randomly selected by the server or be the result of a leadership election protocol. All the clients will then share the same pair of (sk, pk) keys, with the sk key necessary for the decryption and signing and the public key pk for evaluation and verification. The central server holds only the pk required for the homomorphic evaluation and signature of the global model.

At each round of the training which is an iterative process, the server randomly selects K out of the M clients. Each client sends its local updates of the weights homomorphically encrypted together with an authentication tag. The server updates the global model by computing a homomorphic aggregation on the weights. In the same time, it computes the signature tag associated with the global model. The homomorphic result and the signature tag are sent back to the K clients. Each of them will verify that the global model was computed correctly and if so they will decrypt the received global weights. This concludes the current round and a new iteration can begin. If the verification fails, the clients will take appropriate actions, outside the scope of our work.

Once the training is finished, the parameters of the final global model are sent back to the M clients which decrypt them and explore them internally for prediction on their local datasets.

It is noted that the same technique can be extended with minor modifications to the case we consider gradients instead of weights as local updates sent by the clients.

As example of a relevant application of our secure FL framework, one can cite the training of a federated model across multiple medical institutions without sharing the patient

data. This allows to collaborate and build relevant models while keeping the patient data in local, at the hospital and thus reduce the risk of data leakage while respecting the health regulations. Another example of use case is the training of a common model by the partners of a common defence alliance (*e.g.* NATO) without sharing their sensitive military data.

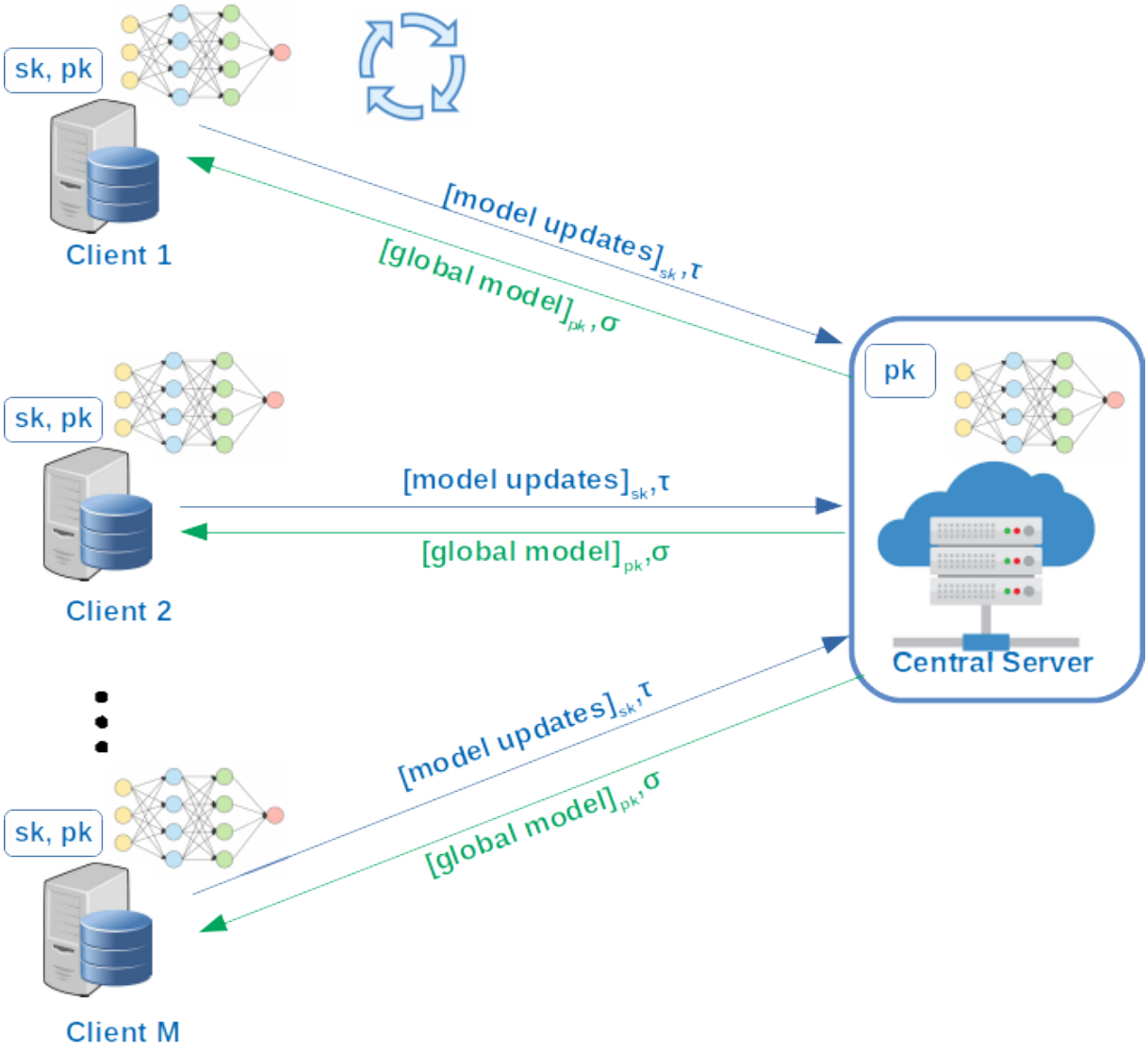

Figure 6.1: Cross-silo federated learning architecture with Homomorphic Encryption and public Verifiability

6.4.2 Threat and security analysis

Following common cybersecurity practices, this section summarizes the security properties of our framework in terms of assets, threats and countermeasures. In our setup we have two assets: the training data, owned by the clients, and the model they collectively build with the help of the aggregation server. First, the confidentiality of the training data of a given client must be guaranteed with respect to threats coming from both the server as well as (ideally) the other clients. Additionally, the confidentiality of the model (or, rather the successive models) must be guaranteed with respect to threats from

the server (as all the clients are granted access to the succession of models, there is no confidentiality requirements on the models w. r. t. the clients in our protocol). The integrity of the model should also be guaranteed against threats coming from the server and, ideally, from the clients. In this setup, the framework presented here encompasses two complementary countermeasures in order to address the above server threats: Fully Homomorphic Encryption and public Verifiable Computing. Since FHE allows the server to perform its aggregation function by working directly over encrypted model data, it addresses confidentiality threats on the model data coming from the server and, as a by-product that the server works in the encrypted-domain, on the clients' training data as well. Now, turning to integrity, our Verifiable Computing-approach allows all clients to formally establish that the server correctly performs its aggregation function (albeit on encrypted data) and, as such, mitigates integrity threats from the server. At present, client threats are not (yet) covered by our framework. In particular Differential Privacy, by adding an appropriate noise to the successive models coefficients (thus preventing model inversion attacks and the like by the clients which receive the successive models), can help mitigating confidentiality threats on the clients training data with respect to one another. Still, properly introducing DP within a FL framework is easier said than done and will be the focus of another work. The last kind of threats that our framework does not fully cover (and which would also stay uncovered even when bringing DP into the picture) concerns integrity threats on the model coming from the clients [160]. It should however be emphasized that these threats are very hard to mitigate as malicious clients can misbehave in many different and possibly harmful ways (*e.g.* from lying on their training set sizes to using false or even misleading training data). Still, some form of integrity on the final model can be checked a posteriori, for example by having each client running the model on its own private test set and measuring the resulting classification rate. Let us also note that our security model is valid under the hypothesis of non collusion between the server and any user participating in the FL protocol.

6.4.3 Cryptographic tools and optimizations

As detailed in the section 6.3.3, one can batch several messages into the same Paillier ciphertext. In the context of our FL approach, the weights updates by client as well as the overall global model parameters are quite important in terms of size. By batching the weights local updates and the weights of the global model one can diminish the bandwidth requirements and also the evaluation time on the server side.

Let us now give the example on how the batching technique applies on federated averaging. For the federated averaging, on the central server side, one must compute in the encrypted domain: $w_{t+1} \longleftarrow \sum_{k=1}^{K} \frac{n_k}{n} * w_{t+1}^k$ based on the encrypted updates of the weights received from the clients k.

Of course $0 < n_k/n \leq 1$ with r representative digits after the decimal point. The weights are usually real numbers for which we will keep only p representative digits of precision. For simplicity reason, we suppose that all weights are translated into the positive domain. In this case, one must have for each slot in the packed message extra space for the term n_k/n. As such, each slot i for a packed ciphertext will be in the form

$$[\underbrace{n_k/n}_{\lceil log_2(10^r) \rfloor} \mid \underbrace{w_{t+1,i}^k}_{\lceil log_2(10^p) \rfloor} \mid \underbrace{0 \ldots 0}_{K}].$$

Therefore, one can pack at most b weights in a single ciphertext with

$$b = \Big\lfloor \frac{log_2(n)}{log_2(10^r) + log_2(10^p) + K} \Big\rfloor.$$

More concretely, let us suppose $n_k/n = (0, r_1r_2r_3)$, each weight is in the form $(0, p_1p_2p_3p_4)$ and $K = 10$. It follows that we can have at most $\lfloor 2048/(log_2(10^3 * 10^4) + K) \rfloor = 61$ packed messages in a single ciphertext.

Let us now go into more details on the training protocol when using the above specified cryptographic primitives.

At each round of the training, each client sends the result of $AEncrypt$ over the local updates of the weights: $c = (C, e, e^{-1}, \mu_e, \sigma)$ containing the ciphers (C, e, e^{-1}, μ_e) concatenated with the corresponding tag σ. The server updates the global model by calling the $AEval$ algorithm over the received messages c and using f, where f is the aggregation function. The output of $AEval$ is sent back to the K clients. Each of them runs the $AVerify$ algorithm to verify that the global model was computed correctly. If so they will evaluate the $ADecrypt$ over the message received to decrypt this message and obtain the global weights. As mentioned in the section 6.3.4, the VC scheme for Paillier encryption (LEPCoV) verifies the outsourced computation of any linear function over any messages in $\mathbb{Z}_{N_E}$. Then to adapt the LEPCoV for the Paillier batching encryption of weights it is sufficient to modify only the message weights w_k to w'_k, the packed version of the weights. Then each user runs $AEncrypt(sk, \tau, i, w'_k)$ instead of $AEncrypt(sk, \tau, i, w_k)$ and the cipher becomes $c = (C', e, e^{-1}, \mu_e, \sigma)$, where $C' = Enc(w'_k)$ as illustrated above. The remaining of the framework is completed like before. Finally, we note that in the evaluation algorithm, we can evaluate C, b, and x in parallel (i.e. the runtime of $AEval = max(AEval(C), AEval(b), AEval(s) + AEval(a) + AEval(x))$). Each user can run $AEncrypt$ and respectively $AVerify$ in parallel over the batches of uploaded and respectively downloaded weights so the associated evaluation times can also be reduced.

6.5 Experimental results

We use the Federated Extended MNIST (FEMNIST) dataset[1] as experimental setup. The extended version of MNIST contains 62 classes (digits, upper and lower letters) and comes with the writer id in such a way that its federated version was built by partitioning the data based on the writer [183]. Among the 3,596 writers contained in the original dataset, we keep the 500 users with the most data. Each selected user's dataset has a train/test data split for a total of 165,050 train and 27,433 test images.

Evaluations were done with a standard CNN composed of two convolution layers (the first with $5 * 5$ kernel size and 128 channels, the second with $3 * 3$ kernel size and 64 channels, each followed with $2 * 2$ max pooling), a fully connected layer with 512 units and ReLu activation, and a final softmax output layer (486,654 total parameters). The evaluation metric was the accuracy on the test sets and, for each experimentation, 200 learning rounds were done.

[1]Dataset available at https://www.nist.gov/itl/products-and-services/emnist-dataset

We have led two distinct sets of experimentation. On one hand, we aimed at finding the best hyperparameters (K, B, E)[2] to improve the speedup of the learning process and thus decrease the communication cost. On the other hand, our concern was about privacy and our goal was to secure the FL process without degrading its performance.

6.5.1 Setting FL hyperparameters

To evaluate the speedup of the learning process relative to the FL hyperparameters, we report the number of communication rounds to reach a decent target accuracy of 80%. We first experiment with the number K of participants per round, which controls the amount of multi-client parallelism. Setting $B = 5$ and $E = 10$, we show the impact of varying K (see Fig.6.2). Above $K = 10$, there is only a small advantage in increasing the client fraction. Thus, for the remainder of our experiments we fix $K = 10$, which strikes a good balance between computational efficiency and convergence rate.

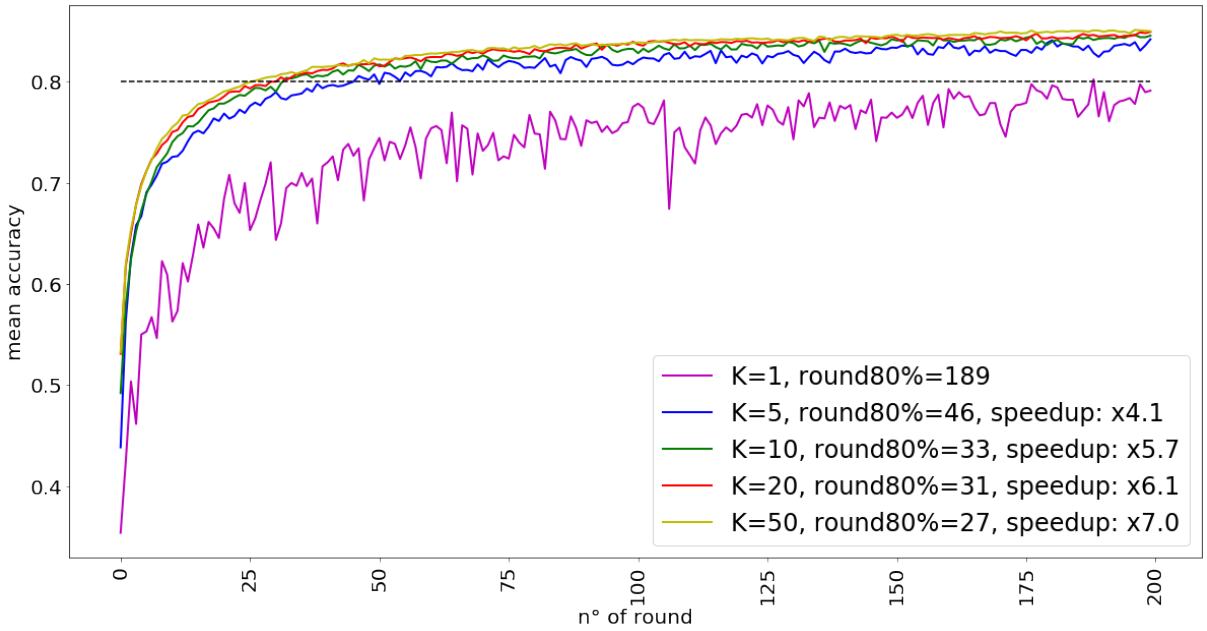

Figure 6.2: Test set accuracy vs. communication rounds varying the number of participants (B=5 and E=10)

Our second experiment aims at choosing B and E, which control the number of local calculations for each client. Starting from $B = 50$ and $E = 1$, we add more computation per client on each round, either decreasing B, increasing E, or both. Table 6.1 demonstrates that adding more local SGD updates per round can produce a dramatic decrease in communication costs. In the following, we use the parameters B=5 with E=10 which give the best convergence rate for the FL process.

6.5.2 Quantization vs. utility

Since the encryption quantifies the clear messages, we conducted experiments to analyze the impact of this quantization on the utility of the model. Contrary to Section 6.5.1,

[2]Notations are those introduced in Algorithm 1

		E			
		1	5	10	20
B	50	-	195	149	152
	10	155	50	48	44
	5	87	40	**33**	34

Table 6.1: Number of communication rounds to reach a target accuracy of 80% varying both B and E (while K=10)

we fixed the number of learning rounds to 200 and reported in Table 6.2 the accuracy varying the precision on both w_k and $\frac{n_k}{n}$ for each participant k. We observe that a 10^4 precision is required not to degrade performances. In Figure 6.3, we focus on performance deterioration due to precision on w_k by showing accuracy for each round considering different weights precisions (while precision on $\frac{n_k}{n}$ is float32).

		precision on w_k			
		float32	10^4	10^3	10^2
precision on $\frac{n_k}{n}$	float32	**84.6%**	**84.2%**	82.6%	75.4%
	10^4	**84.5%**	**84.6%**	82.8%	75.1%
	10^3	83.5%	83.3%	82.0%	75.4%
	10^2	78.0%	78.0%	77.2%	73.9%

Table 6.2: Accuracy after 200 learning rounds depending on precision on both w_k and $\frac{n_k}{n}$

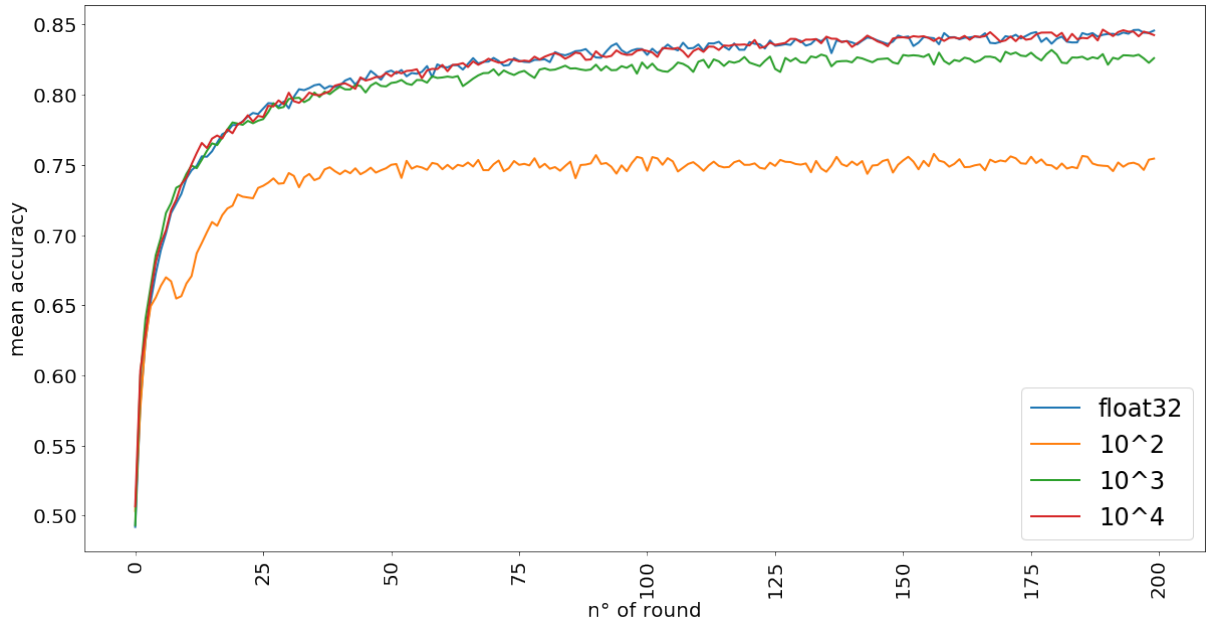

Figure 6.3: Test set accuracy vs. communication rounds varying the precision on weights w_k (while precision on $\frac{n_k}{n}$ is float32)

6.5.3 Performance evaluation of LEPCoV scheme

In the beginning, let us specify that all tests presented in this section were performed on a 2016 DELL PC(Genuine-Intel Core $i7-6600U$, 4 cores at $2.60GHz$ with $16GB$ RAM at $2.13GHz$), on Ubuntu (Linux kernel $4.15.0-91-generic$, with the architecture $x86-64$) as an operating system. Finally we used the C++ language to implement the LEPCoV scheme.

Table 6.3 summarizes the average runtimes of $AKeyGen$, $AEncrypt$, $AVerify$, $ADecrypt$, $AEval$ and the detailed evaluation, over the tag $\sigma = (a, b, s, x)$, and over the ciphers C. Note that the evaluation time of $AEncrypt$ and $ADecrypt$ depends only on the size of the cryptosystem parameters and the evaluation time of $AKeyGen$, $AEval$, $AVerify$ further depends on the number of participants K. We note that the $AKeyGen$ is performed once. The time presented in this table for $AEncrypt$ is the time evaluation for the encryption and tag generation for one message (weight). We recall that we can evaluate f over C, b, and x in parallel (i.e. the runtime of $AEval = max(AEval(C), AEval(b), AEval(s) + AEval(a) + AEval(x))$). We also remark that the verification time is rather fast (e.g $28,87$ms for $K = 10$ and a 2048 bits modulus).

Modulus size	1024			2048			3072		
K	10	20	50	10	20	50	10	20	50
$AkeyGen$	19.98	20.07	21.04	163.9571	167.89	170.84	638.79	644.5	645.44
$AEncrypt$	5.17	5.23	5.36	37.12	37.05	37.12	124.07	117.39	118.6
$AVerify$	4.36	4.54	5.93	28.87	30.3	34.14	96.54	96.21	101.35
$Decrypt$	1.71	1.26	1.21	8.6	9.01	8.69	28.18	27.8	28.34
$AEval$	0.38	0.46	1.16	2.76	1.57	4.16	8.58	8.17	9.31
$AEval(c)$	0.23	0.46	1.14	0.8	1.57	4.08	1.77	3.82	9.31
$AEval(a)$	0.0006	0.0008	0.001	0.0008	0.001	0.002	0.001	0.001	0.003
$AEval(b)$	0.23	0.43	1.16	0.77	1.55	4.16	1.75	3.55	8.53
$AEval(s)$	0.0009	0.001	0.004	0.001	0.001	0.003	0.001	0.002	0.004
$AEval(x)$	0.38	0.37	0.42	2.76	2.73	2.5	8.58	8.17	7.52

Table 6.3: Average runtimes (in ms) of **AkeyGen, AEncrypt, Adecrypt** and **AEval**, for different modulus and data size K. With the function $f(x) = \sum_{i=0}^{k} n_k/n \; w_i$ where n_k/n and w_k of size 10^4

Table 6.4 shows the batching characteristics one can use to encrypt with Paillier cryptosystem the clients updates for the CNN model described earlier with 486,654 parameters. As such, we report the number of slots (column "#slots") and the number of ciphertexts (column "#ctxts") per participant in one round in function of the number of participants (K), the precision of the weights w_k and of the term n_k/n as well as the modulus size. The number of slots (and implicitly the number of encrypted messages per round) depends on the number of bits of the modulus, the number of participants K and the precision of both n_k/n and w_k.

Table 6.5 shows the bandwidth size in one round between a client and the central server on upload and download. On the upload (direction client-server) this message is the output of AEncrypt algorithm, it thus contains the cipher c and the tag of verification σ. On the download, it contains the result of $AEval$ algorithm that runs on the central server.

		Modulus size											
		1024				2048				3072			
	n_k/n precision	10^4		10^3		10^4		10^3		10^4		10^3	
K	w_k precision	#slots	#ctxt	#slots	#ctxt	#slots	#ctxt	#slots	#ctxt	#slots	#ctxt	#slots	#ctxt
10	10^4	27	18024	30	16221	55	8848	61	7977	83	5863	92	5289
	10^3	30	16221	34	14313	61	7977	68	7156	92	5289	102	4771
	10^2	34	14313	38	12806	68	7156	76	6403	102	4771	115	4231
20	10^4	21	23174	23	21158	43	11317	47	10354	65	7486	71	6854
	10^3	23	21158	25	19466	47	10354	51	9542	71	6854	76	6403
	10^2	25	19466	27	18024	51	9542	55	8848	76	6403	83	5863
50	10^4	13	37434	13	37434	26	18717	27	18024	40	12166	41	11869
	10^3	13	37434	14	34761	27	18024	29	16781	41	11869	43	11317
	10^2	14	34761	15	32443	29	16781	30	16221	43	11317	46	10579

Table 6.4: Paillier batching requirements in function of the precision and number of participants

For example, for a modulus size of 2048 bits, 10^4 of precision for both n_k/n and w_k and $K = 10$, each user (out of 10) sends $486,654 \times 5.6$ KB $\simeq 2.5$ GB to central server without batching or it sends 8848×5.6 KB $\simeq 48.3$ MB to the central server with batching. Each client receives the same message from the server of the size 2.5 GB in the case without batching or 49.5 MB with batching. The 5.6 KB mentioned above is the size for one message, containing the size of the ciphers $C = (c, e, e^{-1}, \mu_e, \tau,)$ auditioned with the size of the tag $\sigma = (a, b, x, s)$, both of equal sizes.

		without batching	with batching		
	K / mod	10,20,50	10	20	50
Client, Server	1024	1200	49.2	63.3	102.3
Client, Server	2048	2500	48.3	61.8	102.3
Client, Server	3072	3800	47.5	60.6	98.6

Table 6.5: Size of bandwidth (in MB) between one client and the central server in one round, with 10^4 precision of both n_k/n and w_k.

Table 6.6 shows the *sequential* evaluation times for the different cryptographic primitives with batching, for one round of the training, for a precision level of 10^4 for both n_k/n and w_k and a modulus size of 2048 bits. The experiments were performed for a modulus on 2048 bits since in terms of security, a modulus size of 2048 bits provides long-term security guarantees following common practice, 1024 bits is considered insufficient and 3078 bits is reserved for "beyond 30 years" confidentiality requirements. Columns "unit" report the times per unitary encrypted messages while columns "total" report the times spent to execute the model with all the 486 654 parameters but without any parallelization. Let us however emphasize that the execution times especially for $AEncrypt$ and $AVerify$ can be further improved by involving simple multi-core parallelization, which seems a viable option in the cross-silo FL context to which this work applies. Indeed, on the client side, each ciphertext (which contains several weights) can be prepared independently (and furthermore the Paillier encryption function can be split in a message independent part, which can be precomputed, and an online message dependent part in order to reduce latency). Similarly, on the server side, the averages can also be computed independently. So due to the "embarrassingly parallel" nature of both, the computing times in Table 6.6 can easily (*e.g.* by an OpenMP parallel-for) be reduced by one or two orders of magnitude depending on the number of cores of the machines involved in the protocol.

Again, involving high-end machines on both client and server sides seems realistic in the cross-silo setting.

	$K = 10$		$K = 20$		$K = 50$	
	unit	total	unit	total	unit	total
AEncrypt	0.73	6469.7	0.756	8552.49	0.741	13865.18
AEval	0.03	244.82	0.06	650.50	0.14	2619.63
AEval(c)	0.028	244.82	0.06	650.50	0.14	2619.63
AEval(a)	≈0	0.02	≈0	0.06	≈0	0.18
AEval(b)	0.03	243.94	0.06	636.35	0.14	2619.63
AEval(s)	≈0	0.04	≈0	0.09	≈0	0.19
AEval(x)	≈0	36.45	≈0	46.62	≈0	78.61
AVerify	0.07	579.54	0.10	1144.15	0.21	4004.69
ADecrypt	0.01	76.09	0.01	97.89	0.01	160.97

Table 6.6: *Sequential* runtimes (in seconds) for 10^4 precision of both n_k/n and w_k and a modulus size of 2048 bits

6.6 Conclusion and perspectives

The framework presented in this work addresses both confidentiality and integrity threats, on both the training data and model, coming from the aggregation server by means of HE and VC techniques. On top of providing strong cryptographic security guarantees, and despite the far from negligible overhead induced by these techniques, we claim that our framework achieves practical performances at least in cross-silo setting when the participants are willing to deploy high-end machines (between 10 to 100 cores) to decrease the overall protocol latency to a sustainable level.

As such, this framework is a significant step towards private-by-design federated learning. However, it is also desirable to cover a wider security model by also countering threats from the end-users thus extending our solution to a fully secure framework applicable in a context where the recipients of the final model may not be trusted (to some extent). Notably, this will require to bring differential privacy (DP) into the picture in order to prevent indirect leakage of sensitive information on the training data from the successive models built (and disclosed to the clients) in the protocol. In a scenario in which the clients or the end-users of the final model may be malicious, DP would indeed protect the model updates or the model itself from attacks like membership inference [181, 182] or model inversion [160, 162]. Yet, there are a number of subtleties in doing so, in particular with respect to distributed noise generation or interferences between HE and VC on one hand and DP on the other hand.

Chapter 7

Secure TL using VC and HE

7.1 Introduction

In recent years, a major domain of research concerns the machine learning (ML) methods and the efforts in having high quality predictive models [184–186]. Yet, in practical usage scenario, it is often necessary to evaluate these models in a privacy-preserving fashion, for example by evaluating a model on a server over encrypted data, the inputs or the derived predictions are not disclosed to the server. In this context, this work studies how complex machine learning can be performed securely in practice by combining Fully Homomorphic Encryption (for computing over encrypted data), Verifiable Computing (for integrity guarantees) as well as Transfer Learning (as a means for scaling without prohibitively large volumes of costly encrypted operations). The common point of the most previous works in the Machine Learning domain is that the training data and testing data enjoy precisely the same feature space and identical data distributions. In contrast, Transfer Learning (TL) aims to build an effective model that transfers knowledge in one context to enhance learning in a different context. Therefore, it predicts even if the data distribution is not identical with the previous one, without constructing a new model from scratch. This is interesting for various reasons such as saving efforts, energy, and time.

Classification is one of the most investigated applications of transfer learning [187–198]. The problem of lacking sufficient labeled or unlabeled data in a target domain can be solved by the transfer learning, which also leads to more reliable classification results. Other typical applications exploiting TL have been proposed in recent years, such as pedestrian detection [199], improved image recognition in the medical field [200], improving visual tracking [201], and features selection [202, 203]. In the machine learning context, one important issue is data confidentiality and model integrity. Homomorphic Encryption is one of the methods to ensure the data privacy that allows to apply an operation over encrypted data without decrypting it first. The integrity of computation on encrypted data or clear data can be further on verified using verifiable computing techniques.

In this context, this contribution addresses the confidentiality threat and the leakage of information in the transfer learning process. We leverage homomorphic encryption and verifiable computing to provide solutions for the secure evaluation step of the transfer learning. Our contributions are as follows:

- Propose a secure architecture for privacy-preserving and verifiable TL by means of Homomorphic Encryption and Verifiable Computing. Beside this architecture, we also give an example of a practical medical use case for our secure transfer learning solution.
- Provide an instantiation of our framework using the VC protocol from [3] with the BFV [24] homomorphic encryption scheme, to classify an encrypted image into two classes (dogs and cats)[1].
- Propose PEOLE, a method of dimension reduction of the features space in order to efficiently apply the homomorphic encryption techniques without a significant accuracy loss.
- Evaluate the practical performances of our architecture (≈2 min for prediction an encrypted image) by several classification experiments consisting in extracting the feature from a pre-trained model VGG16 [204] for image classification and train a MLP (Multi-Layer Perceptron) classifier on top of it.

7.2 Related work

It is worth noting that most of the studies in the privacy of data to machine learning consisting of applying homomorphic encryption to machine learning models concentrates on making the inference on private data (*e.g.* CryptoNets [132], TAPAS [141], NED [142]) and not so much on the training phase. Zhu and Wu [205] have fine-studied how to deal with noisy class label problems in the line of research between supervised and unsupervised learning. In another line of research, Yang et al. [206] have studied the cost of learning when the additional tests can be made to future samples.

A particularly interesting application approach using encryption for performing machine learning for Deep Neural Network (DNNs) has been done in [207] where all the images used for training are encrypted using a tailored-made cryptosystem called Tanaka. Sirichotedumrong, Kinoshita, and Kiya (SKK) scheme [208] proposed a privacy-preserving

[1]Dataset available at https://www.kaggle.com/shivamb/cnn-architectures-vgg-resnet-inception-tl/data.

scheme for DNN that encrypts the images (pixel-based image encryption method) under different keys, and allows one to use data augmentation in the encrypted domain. Glyph [171] is another approach, based on homomorphic encryption, allowing to fast and accurately train DNNs on encrypted data by switching between TFHE (Fast Fully Homomorphic Encryption over the Torus) and BGV cryptosystems.

The problem of ensuring privacy and verifiability in a deep learning system is examined in a few works such as SafetyNets [146], and Slalom [176]. However, these schemes (SafetyNets, Slalom) propose only a small variety of activation functions or require additional hardware assistance as in [176]. An interesting architecture proposed by Madi et al. [2] to achieve both confidentiality and integrity for the inference step of a neural network is comprised of three entities: client, server, and operator. Even if similar to ours, in their case, the server executes the first layers privately (using FHE and VC) and it is the operator which is performing on clear the last layers of the NN. As such, they cannot take advantage of a public pre-trained model and moreover, they are obliged to use an adversarial learning model against the leakage of information after decryption on the operator side.

To the best of our knowledge, this is the first work to use the application of the verifiable computing and homomorphic encryption for the transfer learning setting (i.e. verify the integrity of the encrypted prediction results returned by the model).

7.3 Background

7.3.1 Transfer Learning (TL)

Transfer Learning (TL) is the process of learning to solve a problem in a "target" domain using part of the knowledge acquired on the reference problem to solve a similar target problem. In this context, we can distinguish several approaches depending on what, when and how we want to transfer. The application of this idea in ML implies reusing all or part of a model learned on reference data to solve a target problem by re-learning on the target data. This is attractive for several purposes, such as learning about sensitive data, while respecting privacy policies and business requirements and in different domains like the health or the autonomous driving field.

In a Transfer Learning setting, some labeled data $\mathcal{D}_{src}$ are available in a source domain, while only unlabeled data $\mathcal{D}_{tar}$ are available in the target domain. We denote the source domain data as $\mathcal{D}_{src} = \{(x_{src_1}, y_{src_1}), \ldots, (x_{src_{n_1}}, y_{src_{n_1}})\}$, where $x_{src_i} \in \mathbb{R}^m$ is the input data and y_{src_i} is the corresponding label. Furthermore, we indicate the target domain data as $\mathcal{D}_{tar} = \{x_{tar_1}, \ldots, x_{tar_{n_2}}\}$, and, without loss of generality, we suppose that the input x_{tar_i} in $\mathbb{R}^m$. Let $\mathcal{P}(X_{src})$ and $\mathcal{Q}(X_{tar})$ (denoted by $\mathcal{P}$, and $\mathcal{Q}$ in short) being the marginal distributions of X_{src} and X_{tar}, respectively. Generally, they can be different. The task of transfer learning is then to predict the labels y_{tar_i} corresponding to the inputs $x_{tar_i} \in \mathcal{D}_{tar}$.

7.3.2 Homomorphic Encryption (HE)

Fully Homomorphic Encryption (FHE) schemes allow to perform arbitrary computations directly over encrypted data. That is, with a fully homomorphic encryption scheme

E, we can compute $\mathsf{E}(\mathsf{m}_1 + \mathsf{m}_2)$ and $\mathsf{E}(\mathsf{m}_1 \times \mathsf{m}_2)$ from encrypted messages $\mathsf{E}(\mathsf{m}_1)$ and $\mathsf{E}(\mathsf{m}_2)$.

In this section we recall the general principles of the BFV homomorphic cryptosystem [24], which we use in combination with a VC scheme. Since we know in advance the function to be evaluated homomorphically, we can restrain to the somewhat homomorphic version described below.

The biggest problem of Homomorphic Encryption, especially in the homomorphic multiplicative is the size of the ciphertext that is growing exponentially in the number of operations, which can have a great influence on the correctness and the capability of decryption. For this reason for some HE scheme, there exists a relinearisation operation to solve the growth of error rate. We skip the description of the relinearisation step for the BFV since this is not needed for our usage - we evaluate only multi-variate quadratic polynomials of degree at most 2 (i.e. at most 2 multiplications and a modular reduction which can be realized upon decryption).

Let $R = \mathbb{Z}[x]/\Phi_m(x)$ denote the polynomial ring modulo the m-cyclotomic polynomial with $n' = \varphi(m)$. The ciphertexts in the scheme are elements of polynomial ring R_q, where R_q is the set of polynomials in R with coefficients in $\mathbb{Z}_q$. The plaintexts are polynomials belonging to the ring $R_t = R/tR$.

As such, BFV scheme is defined by the following probabilistic polynomial-time algorithms:

BFV.ParamGen(λ): $\rightarrow (n', q, t, \chi_{key}, \chi_{err}, w)$.
It uses the security parameter λ to fix several other parameters such as n', the degree of the polynomials, the ciphertext modulus q, the plaintext modulus t, the error distributions, etc.

BFV.KeyGen$(n', q, t, \chi_{key}, \chi_{err}, w) :\rightarrow (pk, sk, evk)$.
Taking as input the parameters generated in **BFV.ParamGen**, it calculates the private, public and evaluation key. Besides the public and the private keys, an evaluation key is generated to be used during computation on ciphertexts in order to reduce the noise.

BFV.Enc$_{pk}(m) \rightarrow c = (c_0, c_1, c_2 = 0)$
It produces a ciphertext c according to BFV-cryptosystem for a plaintext m using the public key pk.

BFV.Dec$_{sk}(c) :\rightarrow m$
It computes the plaintext m from the ciphertext c, using private key sk.

BFV.Eval$_{pk,evk}(f, c_1, \ldots, c_n) :\rightarrow c$, with $c =$**BFV.Enc**$_{pk}(f(m_1, \ldots, m_n))$, where $c_i =$ **BFV.Enc**$_{pk}(m_i)$, and f has n inputs and has degree at most two.
It allows the homomorphic evaluation of f, gate-by-gate over c_i using the following functions: **BFV.Add**(c_1, c_2) and **BFV.Mul**$_{evk}(c_1, c_2)$.

For further details on this scheme, we refer the reader to the paper [24].

Let us just note that a BFV ciphertext c can be seen as an element in $R_q[y] = \mathbb{Z}/q\mathbb{Z}[X, Y]/\Phi_m(x)$ with a degree at most 2 (i.e., $c = c_0 + c_1 y + c_2 y^2$).

7.3.3 Verifiable Computing (VC)

Verifiable computation VC techniques allow to prove and verify the integrity of computations on authenticated data. A Verifiable Computation scheme is defined as a protocol in which a client (usually weak) has a function f and some data denoted x and delegates to another client (in most cases a server) the computation of $y = f(x)$. Then the same client or another one can receive the result y plus a short proof of its correctness. More in details, a user generates an authentication tag σ_x associated with his/her data x with his/her secret key and the server computes an authentication tag $\sigma_{f,y}$ that certifies the value $y = f(x)$ as an output of the function f. Now, anyone using the verification key (public or secret) can verify y to check that y is indeed the result of $f(x)$.

A VC scheme includes the following algorithms:

1. (PK,SK)←**KeyGen**(f, λ): Taking as input the security parameter λ and a function f, this randomized key generation algorithm generates a public key (that encodes the target function f) used by the server to compute f. It also computes a matching secret key, kept private by the client.

2. (σ_x,τ_x)←**ProbGen**$_{SK}(x)$: The problem generation algorithm uses the secret key SK to encode the input x as a public value σ_x, given to the server to compute with, and a secret value τ_x which is kept private by the client.

3. σ_y ←**Compute**$_{PK}(\sigma_x)$: Using the client's public key and the encoded input, the server computes an encoded version for the function output $y = f(x)$.

4. (acc, y) ←**Verify**$_{SK}(\tau_x,\sigma_y)$: Using the secret key SK and the secret τ_x, this algorithm converts the server output into a bit acc and a string y. If $acc = 1$ we say that the client accepts $y = f(x)$, meaning that the proof is correct, else (i.e. $acc = 0$) we say that the client rejects it.

7.4 Proposed Approach

7.4.1 Our model

We begin by explaining our proposal for an architecture allowing to deploy the secure transfer learning model using homomorphic encryption and verifiable computing. Our architecture is composed of three entities: user, server, and operator. In the following, we describe a high-level view of our TL design with the role of each entity as revealed in the Figure 7.1. Let us denote by f, the global Machine Learning model deployed by our architecture, consisting of n layers and taking as input the data x.

1. The user - owner of some data denoted x - starts the process by applying the first (n-i) layers of the model, i.e. $f_{n-i}(x)$ When the result is calculated, the user encrypts homomorphically $f_{n-i}(x)$, and she generates the associated integrity tag. These encryption data and the associated tag are sent to the server.

2. The server has the task of evaluating in the homomorphic domain, the remaining layers of the neural network over the private data $[f_{n-i}(x)]_{HE}$. Due to the restrictions imposed by the Verifiable Computing protocol used in our approach (i.e. ability to evaluate the correctness of the evaluation of multi-variate polynomials of degree at most two), in our case the server will homomorphically evaluate only a quadratic

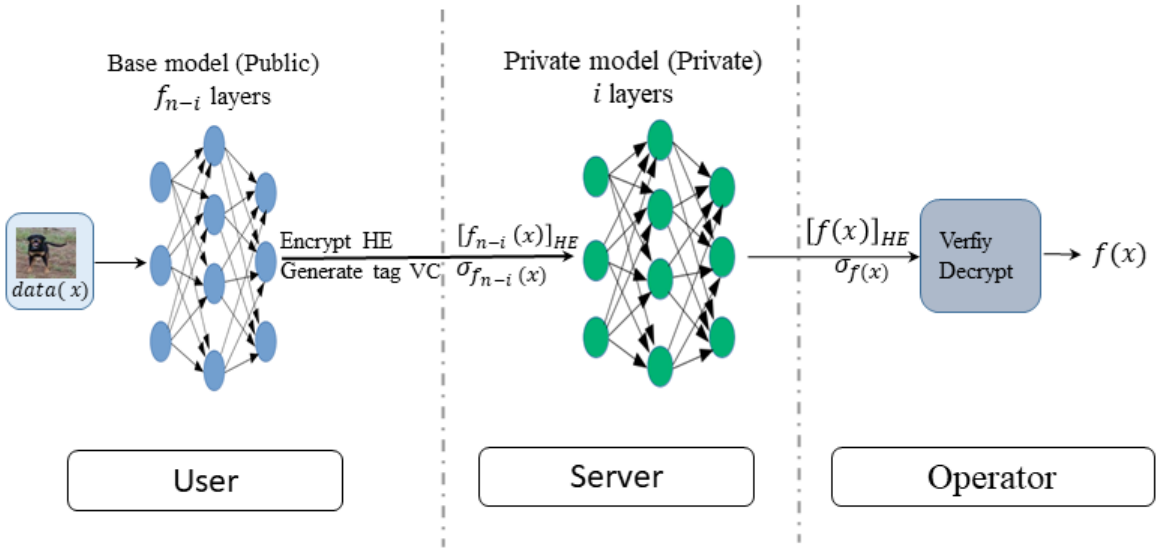

Figure 7.1: Our architecture for a confidentiality & integrity preserving inference phase of transfer learning

function (which is totally feasible and with really good performances by existing FHE means). Now, the homomorphic evaluation of the private part of the model along with the associated integrity proof is sent to the operator.

3. The operator has access to the evaluation of the server evaluation, and to the result of the TL model. Therefore, it can check the validity of the server computation. If it is correct it decrypts the result and then employs it as it wants.

We note that the number of layer i evaluated on the server side, depends directly on the limitation of the VC and FHE methods.

7.4.2 Security Guarantees and Threats

Unlike other works using homomorphic encryption for private inference, we set up our study in the case of a *malicious* server, which can possibly alter the results of the evaluation (e.g. by not running the specified algorithm). We argue that it is necessary to have integrity guarantees against threats coming from this adversary in addition to the confidentiality offered by the homomorphic encryption.The malicious adversaries may conduct arbitrarily (i.e. execute any computation) for stealing, corrupting, and modifying data, without any specifications, and may compute any function over data instead of the required computation (function delegated). For this goal, in our approach, we use verifiable computing technique, in particular the VC protocol of Fiore et al. [3], that allows anyone to check efficiently the calculation evaluated on the server over encrypted data, in order to check that the server correctly calculates the layers delegated to it. To the best of our knowledge, this VC is the most practical verifiable computing protocol to address the validity of computation over encrypted data with the limitation that it evaluates mutivariate functions of degree at most 2[2]. Therefore, this constraint restrains the number of

[2]This is due to the need to go beyond bilinear maps to achieve higher degrees in the underlying cryptographic primitives involved in both VC and FE.

layers that can be delegated to the server in our architecture (i.e. the server can evaluate homomorphically maximum the last quadratic layers of a model).

In our architecture, unlike other works using homomorphic encryption, especially Glyph [171] and Madi et al. work [2], the server evaluates in the homomorphic domain the last layers of the model as knowing the encryption of $f_{n-i}(x)$ to obtain the encryption of $f(x)$. The operator, after receiving the encryption result decrypts $f(x)$, which is contrary to the Glyph and Madi approach, where they compute the first layers of the model and send this encrypted to the operator that decrypts this result and obtain $f_1(x)$ and complete model. Therefore, in terms of information leakage, it is clear that our model is optimal. In summary, using our architecture we prevent threats coming from the server executing the last layer and wanting to infer information about the learning.

7.4.3 Medical Use-Case

As quickly described in [209], imagine a scenario where a radiologist (user) has just acquired images from the body of one of its patients and needs to interact with a remote proprietary diagnostic service (server) to get some insights. The service itself is expecting images as input and crunches them through an advanced deep neural network which outputs highly reliable insights in terms of the pathology the patient is suffering from as well as personalized treatment approaches. Clearly, as health-related data, the patient images and data are considered sensitive and cannot be shared without protecting their confidentiality. On the other hand, the neural network has been carefully crafted by the service provider using a lot of precious hard-earned training data and is considered critical intellectual property. In this scenario, it is thus acceptable neither that the service provider is granted access to the patient data (or to a by-product of these) nor that the radiologist is disclosed the network. Thus without additional means to prevent disclosure of these assets, a high value service is prevented to exist.

One way to resolve these conflicting requirements is by bringing privacy preserving FHE calculations into the picture. In principle, in the above scenario, the radiologist may be the owner of a FHE cryptosystem and send its patient's data encrypted under that cryptosystem to the service provider. The service provider then evaluates its neural network directly over these encrypted data, producing results which are sealed under the radiologist's cryptosystem. The final results are then sent back to the radiologist who is the only party able to decrypt them. So we are done. The patient data are not disclosed to the service provider (since they, and their by-products, are sealed under a cryptographic layer at all time) and the network is not disclosed to the radiologist since it stays on the service provider computing premises. Unfortunately, this naïve view is impractical since, despite the advances made and yet to be made in FHE operators efficiency, it is unlikely that they will be sufficient to enable practical homomorphic evaluation of the large scale neural nets involved in advanced machine learning tasks. Fortunately, as we shall now see, scaling FHE calculations to complex machine learning tasks does not necessarily mean scaling FHE calculations to large scale models.

Now let us assume that the service provider has followed the transfer learning philosophy to build its neural network. As such, its network can thus be split in two parts:

- A first preprocessing network (e.g. VGG16) which is publicly available and has no dependencies on the precious hard-earned training data of the service provider.

- A second much smaller decisional network trained on the service provider sensitive data which turns VGG16 ouputs into the highly reliable insights expected by the radiologist.

In light of the above, we can now rework our scenario to make it much more FHE friendly. Indeed, the publicly available preprocessing network can be disclosed to the radiologist's information system and run in the clear domain before encryption. So rather than sending FHE-encrypted images, the radiologist('s information system) only sends a FHE-encryption of the resulting feature vector(s), which is furthermore of much smaller size than high-resolution images. On the service provider side, only the smaller decisional network has to be run in the encrypted domain therefore dramatically decreasing the footprint of FHE-calculations and resulting de facto in much better scaling properties. Since transfer learning techniques are widely applicable and applied in the neural network community we can therefore claim that performing advanced machine learning tasks over encrypted data does not require scaling encrypted-domain calculation to large networks, as, as argued above, the fact of running the preprocessing on the user/radiologist side does not impact the confidentiality properties of the setup.

7.5 Dimensionality Reduction of target domain

In order to provide a good TL model, that can be used with our architecture, we propose a dimensionality reduction method of the target domain $\mathcal{D}_{tar}$.

The proposed TL dimensionality reduction, which will be called Probability Elimination of Output with Light Effect (PEOLE) uses a projection map $\psi : \mathbb{R}^m \to \mathbb{R}^{m'}$ that eliminates the features extracted by the public model. It consists in finding the minimal dimension of the feature space such that there is not a very high loss of accuracy in the prediction of the final model (i.e. finding m' s.t. $m' = \psi(\mathbb{R}^m)$).

We note by $X \in \mathbb{R}^{n_2 \times m}$ the matrix representing the $\mathcal{D}_{tar}$ where we put the x_{tar_i} in the $i-th$ line of X. We want to find the new matrix $X' \in \mathbb{R}^{n_2 \times m'}$ representing the $\mathcal{D}_{tar}$.

Our method consists in performing two steps:

First step For any column, we calculate the percentage of elements less than a chosen threshold s(s can be for example 10^{-6}).

Second step Eliminate the column that has a percentage bigger than a chosen percentage p (for example for $p = 90\%$, and $s = 10^{-6}$, we eliminate the column that has more than 90% elements smaller than 10^{-6}).

For a percentage p and a sill s, if we delete all the columns of X that have a percentage more than p elements smaller than s, so we do not lose a remarkable amount of accuracy.

7.6 Experimental Evaluation

In our experiments, we want to evaluate our architecture for the case where we extract the features from a pre-trained model for image classification and train a classifier on top of it. We note that the ML implementations in this chapter are done on Google Collab. We use a dataset consisting in images containing only 2 types of animals (dogs and cats)

[3] where the train folder contains 12,500 images for each class. Each image in this folder has the label as part of the filename. The test dataset contains 12,500 images, named according to a numeric id.

As for the public pre-trained model, we used the VGG16 model [204] to extract the data features, which is a convolutional neural network model proposed by K. Simonyan and A. Zisserman [204]. We note that this model is trained over the ImageNet[4], a dataset of 14 million images belonging to 1000 classes. Our evaluation metric was the accuracy of prediction for the test sets.

The VGG16 architecture consists of:

1. A total of 16 layer in which weights and bias parameters are learnt.
2. This network contains a total of 13 convolutional layers with $3 * 3$ kernel size, and increasing numbers of filters corresponding to the layers, with 3 dense layers for classification (comprises of $4096, 4096$, and 1000 nodes each).
3. Each convolutional layers is followed by a ReLu activation, and a final softmax output layer ($25,088$ total parameters).
4. A $2 * 2$ max pooling applied at different steps (after the: 2nd, 4th, 7th, 10th, 13th convolution layer) to obtain the informative features.

We state that, in our experimentation, we do not want to add the last layer of VGG16 architecture, since we add a classifier, essentially a Multilayer perceptron.

On top of the VGG16 model, we build our own private model, a simple neural network - MLP (Multi-Layer Perceptron) to classify over our own dataset (images of dogs and cats). Then, our private model is a MLP composed of one hidden layer with 55 neurons with an identity activation function that trains using the pre-trained features by VGG16. We note that the weight matrix for the first layer is $25,088 \times 55$ and the hidden layer consists of 55×1 weight vector. We recall that the features extracted from VGG16 are in the form of a vector of length $25,088$. Figure 7.2 represents the different steps in our test architecture.

The first step of our experiment is to encrypt the extracted features using VGG16 and to generate the corresponding authentication tags. Afterwards, we evaluate the MLP model over these encrypted data and their tags (the server private computation part) and send the encrypted result with the computed result tag to an operator that can verify (over the result tag) that the calculation is correct, and decrypt this result if the verification passed.

We build two distinct sets of experimentation. On one hand, we aimed at reducing the output of VGG16 of 25088 features to an acceptable size which permits us to encrypt it using a homomorphic encryption system and to improve the speedup of the prediction over encrypted features. Using our PEOLE method we tried to find the best set of parameters (s, p)[5] while preserving the accuracy level. Let us emphasise that our main focus was on the evaluation of the techniques for privacy and integrity (i.e. homomorphic

[3] Dataset available at https://www.kaggle.com/shivamb/cnn-architectures-vgg-resnet-inception-tl/data.

[4] Dataset available at https://www.image-net.org/

[5] Notations are those introduced in section 7.5

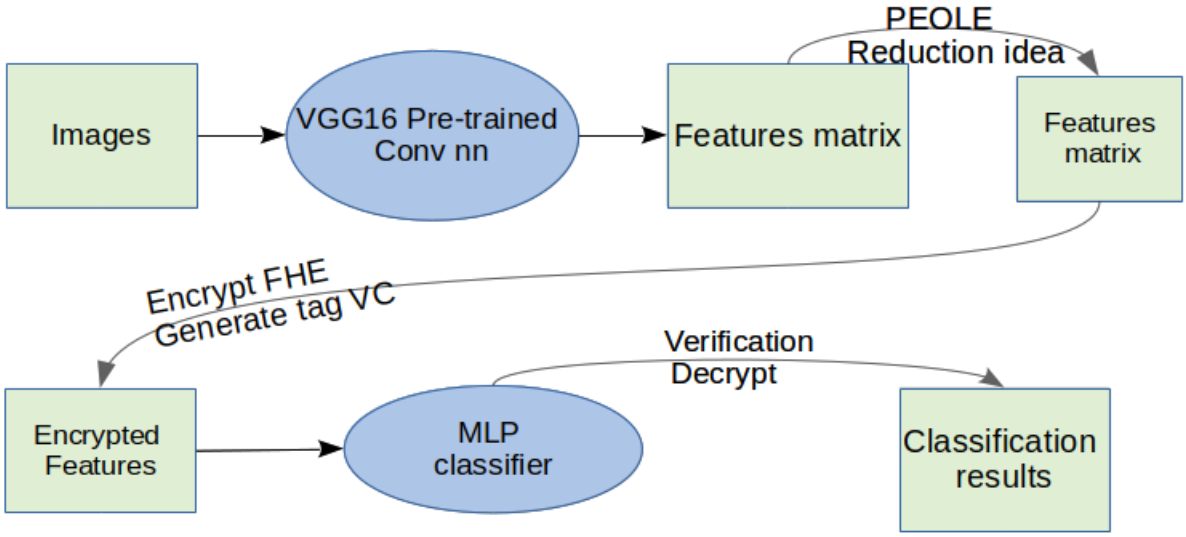

Figure 7.2: Our test architecture.

encryption and respectively VC) in order to achieve a secure transfer learning model without degrading its performance.

7.6.1 Transfer Learning Parameters

To evaluate the speedup of the prediction process relative to the TL parameter and to accelerate the computation of the private model over encrypted data, we start our experiments by describing the variation of the dimension of the target domain (more precisely the output of VGG16) corresponding to the elimination of the column depending on the percentage p of elements with value less than $s = 10^{-2}$. As expected, the dimension of the features vector after elimination is reduced, as seen in Figure 7.3. For example, we remark that the dimension of the output VGG vector is decreased more than 75% for p=80% meaning that its size decreases from 25088 to 5638.

To show a complete view of our method and its importance, we need to describe the accuracy modification using PEOLE with different values for the percentage p. Figure 7.4 describes the evolution of the accuracy with $p \in [80, 100]$ and $s \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. As we can see in this figure the accuracy for $(s, p) = (10^{-2}, 80)$ is 97.5625 for an initial accuracy of 98.3125 (without PEOLE) and then it is a small decrease of accuracy $(0, 75\%)$.We remark that the accuracy can be the same for multiple choices of s and p, e.g. p=$83, 84$ and $s = 10^{-3}, 10^{-4}$ where we obtain an accuracy different from the other s and p values.

As such, PEOLE method is a relatively easy way to explore the features space obtained with the initial public model and to diminish its dimension with a small loss in the accuracy of the final model - i.e. as seen in the Figure 7.4, for $p = 80\%$ and $s = 10^{-2}$, the

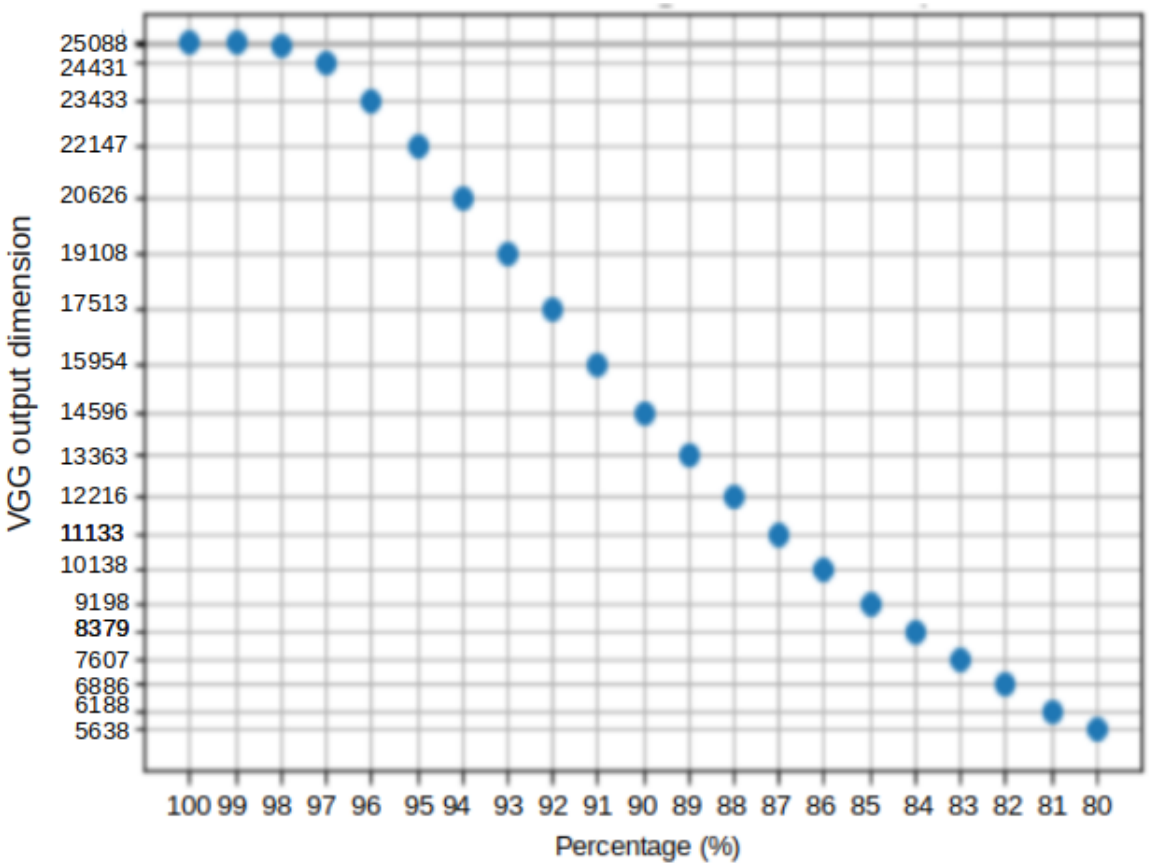

Figure 7.3: Variation of VGG16 output vs percentage using our idea PEOLE.

accuracy becomes 97.5625.

As for the homomorphic encryption, the plaintext are polynomials from the ring $\mathcal{R} =$ with integer coefficients modulo t. Thus one has to encode the features and the parameters of the model before performing homomorphic operations on top of them. As such, we also conducted experiments to analyze the impact of this quantization on the accuracy of the final model. In this test we fixed the parameters s and p to (10^{-2},80%). The accuracy varies depending on the precision of both features(output of VGG16) and the weights of each layer. In Table 7.1, we focus on the performance deterioration due to the approximations on both w_k and f_e by showing the accuracy for the model with regards to the rounding precision for the weights and the features.

Table 7.1 describes the evolution of accuracy depending on the approximation of both: weight and features with fixed $(p, s) = (80\%, 10^{-2})$. We draw your attention that when we refer to a precision of an element of 10^2 for exemple, then we round it to the nearest integer element by taking only the two-digit after the floating point (i.e; for a feature value of 12, 345 the approximated value will be 1234). As you see in this table the accuracy varies from 97.5625% to 97.5 %, then with loss = 0.0625 %, with precision 10^2 of features for any precision of weights, but it is unremarkable variance. For this reason, and taking into account all of the results for the previously mentioned experiments, we work with the following parameters: $s = 10^{-2}$, $p = 80\%$ that produces a feature vector of length 5638, with a good accuracy 97.5 (loss equal 0.8125 %).

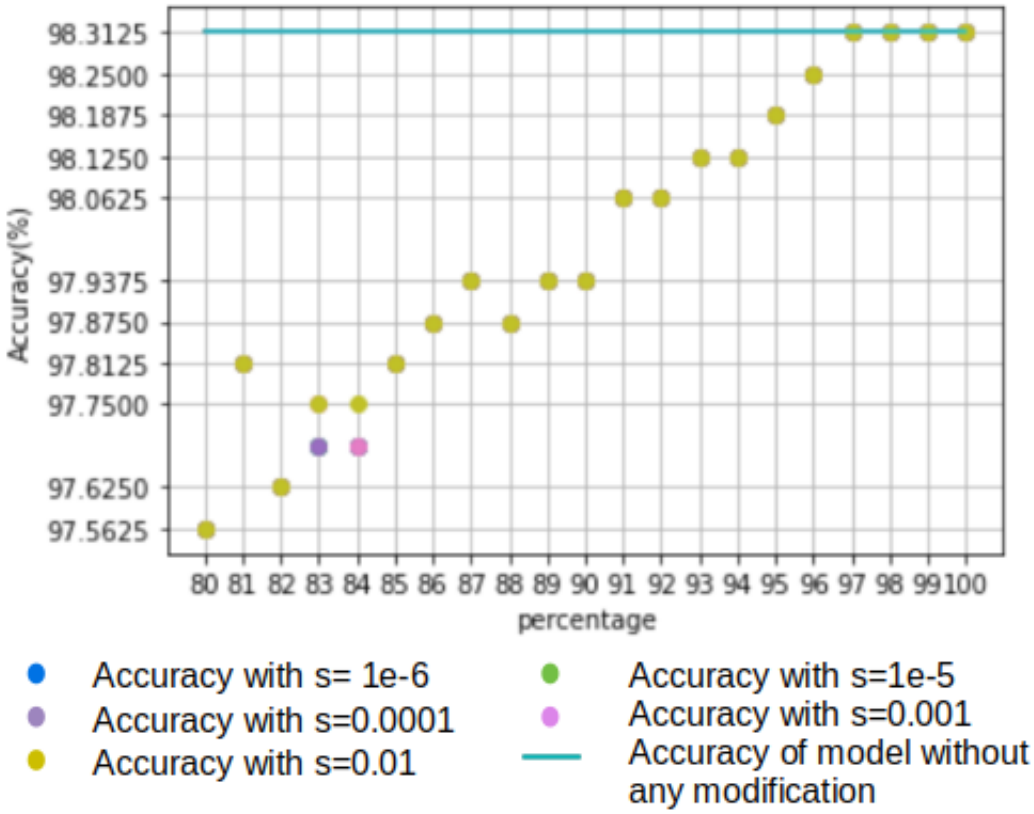

Figure 7.4: Evolution of the model testing accuracy with respect to the percentage p and different s with PEOLE method

		precision on w_i				
		10^2	10^3	10^4	10^5	10^6
precision on fe	10^2	**97.5** %	**97.5**%	**97.5**%	**97.5**%	**97.5**%
	10^3	97.5625 %	97.5625%	97.5625%	97.5625%	97.5625%
	10^4	97.5625 %	97.5625%	97.5625%	97.5625%	97.5625%
	10^5	97.5625 %	97.5625%	97.5625%	97.5625%	97.5625%
	10^5	97.5625 %	97.5625%	97.5625%	97.5625%	97.5625%

Table 7.1: Accuracy of model after application of our PEOLE method with $(p, s) = (80\%, 10^{-2})$ depending on precision on both w_i and fe

7.6.2 Performance of our architecture

In the beginning, let us specify that all tests presented in this section were performed on a 2016 DELL PC(Genuine-Intel Core $i7-6600U$, 4 cores at $2.60GHz$ with $16GB$ RAM at $2.13GHz$), on Ubuntu (Linux kernel $4.15.0-91-generic$, with the architecture $x86-64$) as an operating system. Finally we used the C++ language to implement the encryption and verifiable computing part of our architecture.

In our experiments, we encrypt and decrypt homomorphically the data with the SEAL [154] library and we use the HAL [156] library for authentication but for BFV encrypted data. We note that we choose the security parameters in the way that achieve a 128 bits of security

	operation	times
User-side	Enc	36.11
	$GenTag$	0.005
Server-side	$MLP(Enc(x))$	66.3
	$MLP(\sigma_1, \ldots, \sigma_n)$	1.48
Operator-side	$Verify$	0.027
	Dec	0.002

Table 7.2: Costs (in seconds) for our architecture, where $x = (x_1, \ldots, x_{785})$

Table 7.2 shows the *sequential* evaluation times for the different steps of our architecture to predict the class of one image after applying our PEOLE method for a percentage $p = 80\%$ and a sill $s = 10^{-2}$. Then, the dimension of the features extracted using VGG16 passes from 25088 to 5638. As presented in this table the encryption of this vector of 5638 takes 36.6 *sec* and the generation of the tag for authentication takes 0.005 *sec*. The last one is the cost of generation for a single tag since we note that the tags can be generated in parallel for each encrypted element. The application of MLP over this encryption data takes about one minute and the execution of MLP over the tag takes about 2 seconds (1.48 *sec*) also note that the evaluation of the MLP model and the tag can be executed in parallel. Finally, the verification of authentication the result is very fast and it takes about 0.027 *sec*.

7.7 Conclusion and Future Work

The architecture proposed in this chapter is the first in the TL domain to address both integrity and confidentiality threats by means of homomorphic and verifiable computing techniques. To validate our approach, we evaluated for the evaluation of the last layer of a ML model using unencrypted weights and encrypted feature data. In this scenario, we prevent the threats coming from the server executing the last layer and wanting information about the learning, potentially interested in infering. We create our architecture using the Keras [210] library to build the VGG16 trained over the Imagenet, and test our approach using the MLP with one hidden layer of 55 neurons that show a good accuracy for the private prediction of the class for one image ($\sim$ 97 %) in an acceptable time ($\sim$ 2 min=(2min for HE and in parallel 1.512 min for VC)) with a fast verification of result.

Our architecture remains generic and can be easily extended to further deployments where the private evaluation of the neural network model delegated to the server is more complex (more layers, other activation functions, etc). In order to go further, there are several interesting directions to follow. First, a concrete optimization idea consists in the use of the VC protocols for the homomorphic schemes in batched mode which can improve the performance and reduce the memory used for the encryption data. Other idea is to use newer and more complex verifiable computing protocols, in order to be able to evaluate more than quadratic multivariate polynomials. Finally, we hope that this contribution opens the door to further work covering a more general threat model for the secure AI applications using homomorphic encryption.

Chapter 8

Conclusion

8.1 Motivation and Problem Statement

As predicted by Jeremy Rifkin, we assist today at what it is called The Third Industrial Revolution, thanks to the development of more and more online resources and services. The cornerstone of this revolution is personal data, which can be private or not. For this reason, the study of improving and evolving the encryption tools and cryptosystems takes an important place in the last decade, particularly on tools that preserve confidentiality and integrity such as: homomorphic encryption (HE) and verifiable computing (VC).

The HE evolved in the last ten years from a theoretical idea to existence of several cryptosystems, libraries, and a lot of results, software, and hardware, with each of them, focused on a practical side. It is a very active field, with every year bringing performance improvements and new emerging applications.

Verifiable computing has been studied because it is important in cloud computing. Cloud computing has become the leading trend of modern computing since it has highly diverse usages and various final clients: small businesses, hand-held devices and, private users. This main tool used -VC over encrypted data using HE- stated in chapter 4, it boosts the security of outsourced computation and more precisely the execution integrity.

These two cryptographic tools compose the pillars of our work, in the sense of ensuring integrity and confidentiality properties for ML based applications. The state-of-the art is a moving field and new approaches continue to improve their efficiency and effectiveness.

Thus, we started our manuscript by explaining our context, by showing the ML phases with detailed security analysis, specifically on the side of confidentiality threats and integrity threats. We presented an analysis of the existing works. We have shown that the designs of the second generation of FHE schemes (BGV, BFV,...) are at the center of the work of the scientific community. There is also an interest for the schemes of the third generation which have better management of the noise. We presented in details the design of the HE schemes used in our contribution. Moreover, we presented the design of the VC scheme of Fiore et al. [3] that is one of the main pillars for our building of our contributions.

Regardless of the limitation of these tools, they are an important addition for any work where they operate over it. Specifically, these tools offer: good security properties against threats and flexibility of computing over encrypted data. In the ML context, these techniques (i.e. FHE, VC) help with security problems related to the building of good AI models. Indeed, our work demonstrates that scaling confidential and verifiable encrypted domain calculations to complex machine learning functions does not necessarily require scaling these techniques to large volumes of encrypted domain calculations.

These are the motivations that gave us the enthusiasm and passion to dive into this thesis, i.e. using FHE and VC to ensure secure use of Machine Learning.

8.1.1 Our Contribution

We started our PhD by studying the way to eliminate the limitations of existing VC protocols. More precisely, we started by analyzing and trying to generalize the VC protocol of Fiore et al. [3] that is restricted by the delegation of multivariate polynomials of degree of max 2 for data encrypted with BGV homomorphic scheme. The Fiore et al. [3] core tools (homomorphic hash functions and amortized closed-form efficient PRFs) were instantiated with bilinear groups. We work to instantiate it with a multilinear map for more general functionalities, and use these generalized tools to construct verifiable computing schemes on encrypted data for evaluation of multivariate polynomials of higher degree. Still, we encountered another obstacle which is: proposed homomorphic hash function does not work correctly due to the reduction modulo $\phi(x)$ (the modulo in the ciphertext). unfortuanetly, it is a complicated research subject which may constitute a standalone PhD topic. Therefore, taking into account the scope of our thesis, we decided not to dive up into this subject further on. Yet, we obtain from this research the following result: the adaptation of the VC Fiore scheme with BFV homomorphic encryption scheme, that we used to fulfill our goal.

The first contribution of this thesis is the adaptation of the existing FHE and VC tools with their limitations to ensure a *secure evaluation of a neural network* over secure data during an inference phase. We build an architecture allowing the evaluation of a neural network over private data, while conserving the security of user's data. The server evaluates the first layer of NN in the homomorphic domain, and sends the result to an operator that performs the remaining layers with assuring that the server evaluations are correct.

The second contribution involves the training phase of a ML algorithm, known to be difficult for applying homomorphic encryption techniques. As such, we take a deep look at Federated Learning (FL) which is collaborative training allowing to keep local data secure and train over it using a central server. The central server performs only an aggregation function over the locally trained models. We constructed a *FL framework with a high-security level* with both integrity and confidentiality guarantees, in the sense of preserving the confidentiality of participants' local data, and the integrity of the computation made by the central server in each round of the FL algorithm.

As a third major result, we proposed the first *secure Transfer Learning architecture* that eliminates the information leakage issue in the first contribution while preserving functionality as well as integrity and confidentiality guarantees.

One of the advantages of the proposed architectures is that they can be adapted with

the new VC protocols that work over the FHE schemes, that ensure the integrity of computation for multi-variate polynomials of a higher degree.

8.2 Perspective & Future Work

What is next? In this section, we want to answer this question, in the sense, what are the doors that this work opens in the academic or industrial research. There are a lot of approaches to provide ML usage with confidentiality and integrity guarantees by means of hardware or software solutions.

After everything that we have presented in this thesis, it is evident that the **Generalization of VC Fiore scheme** [3] is part of our perspectives. However, we did not have time to finalize this study during this PhD. If done, this generalization allows anyone to safely evaluate a ML model in any phase of machine learning (learning and / or inference phase) by a malicious server, that is to say, ensuring both the confidentiality and the integrity of this delegation.

Another challenge consists of **discovering and/or adapting the existing VC schemes to work with batched homomorphic encryption** that achieves integrity and confidentiality with optimal time and memory uses for this evaluation. We consider this important because the cost of cloud usage is increasing from day to day. The hurdle here is to batch several plaintext messages in one plaintext and at the same time generate a tag. This tag permits the verification of function evaluation over the batched message.

Yet another challenge is **adapting or discovering a VC scheme that can work with another type of Homomorphic encryption** scheme, that can be more versatile than BGV and BFV like TFHE. Depending on the targeted application, this can result in a more efficient and less memory consuming homomorphic execution.

Another interesting perspective is the study of the combination of the Multi-Party Computation with public Verifiable Computing to **propose an efficient Multi-party VC scheme**, essentially in order to use sensitive data from different sources with a guarantee against confidentiality and integrity threats.

Finally, we note that the perspectives are manifold to ensure the integrity for homomorphic computation: for example using techniques other than VC such as Blockchain, hardware based solutions, etc.

Bibliography

[1] David M Mandelbaum. "On a Class of Arithmetic Codes and a Decoding Algorithm Proof: If t 5 r, then M $/C mod M, has exactly t nonzero". In: *IEEE Transactions on Information Theory* (1976) (page 8).

[2] Abbass Madi, Renaud Sirdey, and Oana Stan. "Computing Neural Networks with Homomorphic Encryption and Verifiable Computing". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2020, pp. 295–317 (pages 9, 99, 103).

[3] 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*. 2014, pp. 844–855 (pages 9, 26, 49–51, 53–59, 64, 67, 68, 73–75, 77, 98, 102, 111–113).

[4] Théo Ryffel, Edouard Dufour Sans, Romain Gay, Francis Bach, and David Pointcheval. "Partially encrypted machine learning using functional encryption". In: *arXiv preprint arXiv:1905.10214* (2019) (pages 9, 24, 64, 66, 69, 79).

[5] Eberhard Hechler, Martin Oberhofer, and Thomas Schaeck. "The operationalization of AI". In: *Deploying AI in the Enterprise*. Springer, 2020, pp. 115–140 (page 16).

[6] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. "Applied cryptography". In: *CRC, Boca Raton* (1996) (pages 17, 20).

[7] Martin Zuber. "Contributions to data confidentiality in machine learning by means of homomorphic encryption". PhD thesis. Université Paris-Saclay, 2020 (pages 18, 19).

[8] Arvind Narayanan and Vitaly Shmatikov. "How to break anonymity of the netflix prize dataset". In: *arXiv preprint cs/0610105* (2006) (page 22).

[9] Pierangela Samarati and Latanya Sweeney. "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression". In: (1998) (page 22).

[10] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. "l-diversity: Privacy beyond k-anonymity". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), 3–es (page 23).

[11] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. "t-closeness: Privacy beyond k-anonymity and l-diversity". In: *2007 IEEE 23rd International Conference on Data Engineering*. IEEE. 2007, pp. 106–115 (page 23).

[12] Klara Stokes and Vicenç Torra. "n-Confusion: a generalization of k-anonymity". In: *Proceedings of the 2012 Joint EDBT/ICDT Workshops*. 2012, pp. 211–215 (page 23).

[13] Cynthia Dwork. "Differential privacy". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2006, pp. 1–12 (page 23).

[14] Stanley L Warner. "Randomized response: A survey technique for eliminating evasive answer bias". In: *Journal of the American Statistical Association* 60.309 (1965), pp. 63–69 (page 23).

[15] C Dwork and A Roth. *The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 9 (3/4): 211–407.* 2014 (page 23).

[16] Damien Desfontaines and Balázs Pejó. "Sok: differential privacies". In: *arXiv preprint arXiv:1906.01337* (2019) (page 23).

[17] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. "On data banks and privacy homomorphisms". In: *Foundations of secure computation* 4.11 (1978), pp. 169–180 (pages 24, 33).

[18] RL Rivest, A Shamir, and L Adleman. "A method for obtaining digital signatures and publi-key cryptosystems, Communications of the ACM 21". In: (1978) (pages 24, 29).

[19] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 169–178 (pages 24, 29, 32, 33, 49).

[20] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis. Stanford, CA, USA, 2009. isbn: 9781109444506, 2009 (pages 24, 33).

[21] RL Rivest. "Shamir, a. and Adelman". In: *L." On Digital Signatures and Public Key* (1978) (page 24).

[22] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *International conference on the theory and applications of cryptographic techniques*. Springer. 1999, pp. 223–238 (pages 24, 29, 33, 39, 40, 48, 85).

[23] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping". In: *ACM Transactions on Computation Theory (TOCT)* 6.3 (2014), pp. 1–36 (pages 24, 40, 42, 50).

[24] Junfeng Fan and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption." In: *IACR Cryptology ePrint Archive* 2012 (2012), p. 144 (pages 24, 34, 42, 44, 64, 65, 70, 71, 98, 100).

[25] Adi Shamir. "Identity-based cryptosystems and signature schemes". In: *Workshop on the theory and application of cryptographic techniques*. Springer. 1984, pp. 47–53 (page 24).

[26] Dan Boneh, Amit Sahai, and Brent Waters. "Functional encryption: Definitions and challenges". In: *Theory of Cryptography Conference*. Springer. 2011, pp. 253–273 (pages 24, 56).

[27] Tilen Marc, Miha Stopar, Jan Hartman, Manca Bizjak, and Jolanda Modic. "Privacy-enhanced machine learning with functional encryption". In: *European Symposium on Research in Computer Security*. Springer. 2019, pp. 3–21 (page 24).

[28] 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: *International Conference on Security and Cryptography for Networks*. Springer. 2020, pp. 525–545 (page 24).

[29] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. "Efficient functional encryption for inner-product values with full-hiding security". In: *International Conference on Information Security*. Springer. 2016, pp. 408–425 (page 25).

[30] Edouard Dufour-Sans, Romain Gay, and David Pointcheval. "Reading in the dark: Classifying encrypted digits with functional encryption". In: *Cryptology ePrint Archive* (2018) (page 25).
[31] Hallam Stevens. "Hans Peter Luhn and the birth of the hashing algorithm". In: *IEEE Spectrum* 55.2 (2018), pp. 44–49 (page 25).
[32] Ronald Rivest. *RFC1321: The MD5 message-digest algorithm*. 1992 (page 25).
[33] FIPS PUB DRAFT. "202. SHA-3 Standard: Permutation-Based hash and extendable-output functions". In: *Information Technology Laboratory, National Institute of Standards and Technology. Recovered on May* (2014) (page 25).
[34] William W Plummer. "TCP checksum function design". In: *ACM SIGCOMM Computer Communication Review* 19.2 (1989), pp. 95–101 (page 25).
[35] Dilip V. Sarwate. "Computation of cyclic redundancy checks via table look-up". In: *Communications of the ACM* 31.8 (1988), pp. 1008–1013 (page 25).
[36] Michael Steiner, Gene Tsudik, and Michael Waidner. "Diffie-Hellman key distribution extended to group communication". In: *Proceedings of the 3rd ACM Conference on Computer and Communications Security*. 1996, pp. 31–37 (page 25).
[37] Eli Biham and Rafi Chen. "Near-collisions of SHA-0". In: *Annual International Cryptology Conference*. Springer. 2004, pp. 290–305 (page 25).
[38] Antoine Joux. "Collisions for SHA-0". In: *CRYPTO 2004 rump session (Aug.)* (2004) (page 25).
[39] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. "Cryptanalysis of the Hash Functions MD4 and RIPEMD". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2005, pp. 1–18 (page 25).
[40] Xiaoyun Wang and Hongbo Yu. "How to break MD5 and other hash functions". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2005, pp. 19–35 (page 25).
[41] Secure Hash Standard and PUB FIPS. "180-2". In: *August* 1 (2002), p. 72 (page 25).
[42] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The knowledge complexity of interactive proof systems". In: *SIAM Journal on computing* 18.1 (1989), pp. 186–208 (pages 25, 46).
[43] Joan Boyar, Gilles Brassard, and René Peralta. "Subquadratic zero-knowledge". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1169–1193 (page 25).
[44] Ronald Cramer and Ivan Damgård. "Linear zero-knowledge—A note on efficient zero-knowledge proofs and arguments". In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. 1997, pp. 436–445 (page 25).
[45] Joe Kilian and Erez Petrank. "An efficient noninteractive zero-knowledge proof system for NP with general assumptions". In: *Journal of Cryptology* 11.1 (1998), pp. 1–27 (page 25).
[46] Joan Boyar, Ivan Damgård, and René Peralta. "Short non-interactive cryptographic proofs". In: *Journal of Cryptology* 13.4 (2000), pp. 449–472 (page 25).
[47] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect non-interactive zero knowledge for NP". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2006, pp. 339–358 (page 25).
[48] Yael Tauman Kalai and Ran Raz. "Succinct non-interactive zero-knowledge proofs with preprocessing for LOGSNP". In: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE. 2006, pp. 355–366 (page 25).

[49] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-knowledge from secure multiparty computation". In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 21–30 (page 25).

[50] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. "Signing a linear subspace: Signature schemes for network coding". In: *International Workshop on Public Key Cryptography*. Springer. 2009, pp. 68–87 (pages 25, 50).

[51] Craig Gentry and Daniel Wichs. "Separating succinct non-interactive arguments from all falsifiable assumptions". In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 2011, pp. 99–108 (pages 25, 48).

[52] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. "Pinocchio: Nearly practical verifiable computation". In: *2013 IEEE Symposium on Security and Privacy*. IEEE. 2013, pp. 238–252 (pages 25, 66).

[53] Srinath Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J Blumberg, and Michael Walfish. "Taking proof-based verified computation a few steps closer to practicality". In: *21st {USENIX} Security Symposium ({USENIX} Security 12)*. 2012, pp. 253–268 (page 25).

[54] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. "Quadratic span programs and succinct NIZKs without PCPs". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2013, pp. 626–645 (pages 25, 48).

[55] Michael Backes, Dario Fiore, and Raphael M Reischuk. "Verifiable delegation of computation on outsourced data". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 863–874 (pages 25, 55, 56, 72).

[56] Rosario Gennaro, Craig Gentry, and Bryan Parno. "Non-interactive verifiable computing: Outsourcing computation to untrusted workers". In: *Annual Cryptology Conference*. Springer. 2010, pp. 465–482 (pages 25, 26, 49–51).

[57] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. "How to run turing machines on encrypted data". In: *Annual Cryptology Conference*. Springer. 2013, pp. 536–553 (pages 25, 26).

[58] Dario Fiore, Anca Nitulescu, and David Pointcheval. *Boosting verifiable computation on encrypted data*. 2020 (pages 26, 50).

[59] C. E. Shannon. "Communication theory of secrecy systems". In: *The Bell System Technical Journal* 28.4 (1949), pp. 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x (page 30).

[60] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption & 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 (pages 30, 33).

[61] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption". In: *Journal of computer and system sciences* 28.2 (1984), pp. 270–299 (page 30).

[62] D Dolev, C Dwork, and M Naor. *Non-Malleable Cryptography. STOC'91*. 1991 (page 30).

[63] Yodai Watanabe, Junji Shikata, and Hideki Imai. "Equivalence between semantic security and indistinguishability against chosen ciphertext attacks". In: *International Workshop on Public Key Cryptography*. Springer. 2003, pp. 71–84 (page 31).

[64] 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 (page 32).

[65] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. "Chosen-ciphertext secure fully homomorphic encryption". In: *IACR International Workshop on Public Key Cryptography*. Springer. 2017, pp. 213–240 (page 32).

[66] Frederik Armknecht, Stefan Katzenbeisser, and Andreas Peter. "Group homomorphic encryption: characterizations, impossibility results, and applications". In: *Designs, codes and cryptography* 67.2 (2013), pp. 209–232 (page 32).

[67] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. "Evaluating 2-DNF formulas on ciphertexts". In: *Theory of cryptography conference*. Springer. 2005, pp. 325–341 (page 33).

[68] Nigel P Smart and Frederik Vercauteren. "Fully homomorphic encryption with relatively small key and ciphertext sizes". In: *International Workshop on Public Key Cryptography*. Springer. 2010, pp. 420–443 (page 33).

[69] Craig Gentry and Shai Halevi. "Implementing gentry's fully-homomorphic encryption scheme". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2011, pp. 129–148 (page 33).

[70] Zvika Brakerski and Vinod Vaikuntanathan. "Efficient fully homomorphic encryption from (standard) LWE". In: *SIAM Journal on Computing* 43.2 (2014), pp. 831–871 (page 33).

[71] Zvika Brakerski and Vinod Vaikuntanathan. "Fully homomorphic encryption from ring-LWE and security for key dependent messages". In: *Annual cryptology conference*. Springer. 2011, pp. 505–524 (page 33).

[72] Zvika Brakerski. "Fully homomorphic encryption without modulus switching from classical GapSVP". In: *Annual Cryptology Conference*. Springer. 2012, pp. 868–886 (pages 34, 42).

[73] Craig Gentry, Amit Sahai, and Brent Waters. "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based". In: *Annual Cryptology Conference*. Springer. 2013, pp. 75–92 (page 34).

[74] Jacob Alperin-Sheriff and Chris Peikert. "Faster bootstrapping with polynomial error". In: *Annual Cryptology Conference*. Springer. 2014, pp. 297–314 (page 34).

[75] Zvika Brakerski and Vinod Vaikuntanathan. "Lattice-based FHE as secure as PKE". In: *Proceedings of the 5th conference on Innovations in theoretical computer science*. 2014, pp. 1–12 (page 34).

[76] Léo Ducas and Daniele Micciancio. "FHEW: bootstrapping homomorphic encryption in less than a second". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2015, pp. 617–640 (page 34).

[77] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds". In: *international conference on the theory and application of cryptology and information security*. Springer. 2016, pp. 3–33 (page 34).

[78] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. "Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2017, pp. 377–408 (page 34).

[79] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Journal of the ACM (JACM)* 56.6 (2009), pp. 1–40 (pages 36, 37).
[80] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2010, pp. 1–23 (pages 37, 38).
[81] Miklós Ajtai. "Generating hard instances of lattice problems". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 99–108 (page 37).
[82] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical hardness of learning with errors". In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 2013, pp. 575–584 (page 37).
[83] Oded Regev. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: STOC '05. New York, NY, USA: Association for Computing Machinery, 2005, pp. 84–93. ISBN: 1581139608. DOI: 10.1145/1060590.1060603. URL: https://doi.org/10.1145/1060590.1060603 (page 37).
[84] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings". In: *Journal of the ACM (JACM)* 60.6 (2013), pp. 1–35 (page 37).
[85] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. "Fast cryptographic primitives and circular-secure encryption based on hard learning problems". In: *Annual International Cryptology Conference*. Springer. 2009, pp. 595–618 (page 37).
[86] Adeline Langlois and Damien Stehlé. "Hardness of decision (R) LWE for any modulus". In: *IACR Cryptol. ePrint Arch.* 2012 (2012), p. 91 (page 37).
[87] Martin R Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of learning with errors". In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203 (page 38).
[88] Rachel Player. "Parameter selection in lattice-based cryptography". PhD thesis. Royal Holloway, University of London, 2018 (page 38).
[89] 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* 13.1 (2019), pp. 47–67 (page 39).
[90] Zvika Brakerski and Vinod Vaikuntanathan. *Efficient Fully Homomorphic Encryption from (Standard) LWE*. Cryptology ePrint Archive, Report 2011/344. https://ia.cr/2011/344. 2011 (page 40).
[91] David P Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. "SETI@ home: an experiment in public-resource computing". In: *Communications of the ACM* 45.11 (2002), pp. 56–61 (page 46).
[92] David P Anderson. "Volunteer computing: the ultimate cloud". In: *XRDS: Crossroads, The ACM Magazine for Students* 16.3 (2010), pp. 7–10 (page 46).
[93] Ran Canetti, Ben Riva, and Guy N Rothblum. "Practical delegation of computation using multiple servers". In: *Proceedings of the 18th ACM conference on Computer and communications security*. 2011, pp. 445–454 (page 46).
[94] Michael Walfish and Andrew J Blumberg. "Verifying computations without reexecuting them". In: *Communications of the ACM* 58.2 (2015), pp. 74–84 (page 46).
[95] László Babai. "Trading group theory for randomness". In: *Proceedings of the seventeenth annual ACM symposium on Theory of computing*. 1985, pp. 421–429 (page 46).

[96] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles". In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. STOC '08. Victoria, British Columbia, Canada: Association for Computing Machinery, 2008, pp. 113–122. ISBN: 9781605580470. DOI: 10.1145/1374376.1374396. URL: https://doi.org/10.1145/1374376.1374396 (page 47).

[97] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. "Practical verified computation with streaming interactive proofs". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 2012, pp. 90–112 (page 47).

[98] Justin Thaler. "Time-optimal interactive proofs for circuit evaluation". In: *Annual Cryptology Conference*. Springer. 2013, pp. 71–89 (pages 47, 66).

[99] Sanjeev Arora and Shmuel Safra. "Probabilistic checking of proofs: A new characterization of NP". In: *Journal of the ACM (JACM)* 45.1 (1998), pp. 70–122 (page 47).

[100] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. "Short PCPs verifiable in polylogarithmic time". In: *20th Annual IEEE Conference on Computational Complexity (CCC'05)*. IEEE. 2005, pp. 120–134 (page 47).

[101] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. "Robust PCPs of proximity, shorter PCPs, and applications to coding". In: *SIAM Journal on Computing* 36.4 (2006), pp. 889–974 (page 47).

[102] Joe Kilian. "A note on efficient zero-knowledge proofs and arguments". In: *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*. 1992, pp. 723–732 (page 47).

[103] Silvio Micali. "Computationally sound proofs". In: *SIAM Journal on Computing* 30.4 (2000), pp. 1253–1298 (pages 47, 48).

[104] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. "Efficient arguments without short PCPs". In: *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*. IEEE. 2007, pp. 278–291 (pages 47, 48).

[105] Srinath TV Setty, Richard McPherson, Andrew J Blumberg, and Michael Walfish. "Making argument systems for outsourced computation practical (sometimes)." In: *NDSS*. Vol. 1. 9. 2012, p. 17 (page 48).

[106] Srinath Setty, Andrew J Blumberg, and Michael Walfish. "Toward practical and unconditional verification of remote computations". In: *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems, HotOS*. Vol. 13. 2011, pp. 29–29 (page 48).

[107] Srinath Setty, Benjamin Braun, Victor Vu, Andrew J Blumberg, Bryan Parno, and Michael Walfish. "Resolving the conflict between generality and plausibility in verified computation". In: *Proceedings of the 8th ACM European Conference on Computer Systems*. 2013, pp. 71–84 (page 48).

[108] Amos Fiat and Adi Shamir. "How to prove yourself: Practical solutions to identification and signature problems". In: *Conference on the theory and application of cryptographic techniques*. Springer. 1986, pp. 186–194 (page 48).

[109] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 2012, pp. 326–349 (page 48).

[110] Giovanni Di Crescenzo and Helger Lipmaa. "Succinct NP proofs from an extractability assumption". In: *Conference on Computability in Europe*. Springer. 2008, pp. 175–185 (page 48).

[111] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. "Geppetto: Versatile verifiable computation". In: *2015 IEEE Symposium on Security and Privacy*. IEEE. 2015, pp. 253–270 (page 48).

[112] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. "Succinct non-interactive zero knowledge for a von Neumann architecture". In: *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 2014, pp. 781–796 (page 48).

[113] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. "How to delegate and verify in public: Verifiable computation from attribute-based encryption". In: *Theory of Cryptography Conference*. Springer. 2012, pp. 422–439 (page 48).

[114] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. "Attribute-based encryption for fine-grained access control of encrypted data". In: *Proceedings of the 13th ACM conference on Computer and communications security*. 2006, pp. 89–98 (page 49).

[115] Andrew C Yao. "Protocols for secure computations". In: *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE. 1982, pp. 160–164 (page 49).

[116] Andrew Chi-Chih Yao. "How to generate and exchange secrets". In: *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE. 1986, pp. 162–167 (page 49).

[117] Kai-Min Chung, Yael Kalai, and Salil Vadhan. "Improved delegation of computation using fully homomorphic encryption". In: *Annual Cryptology Conference*. Springer. 2010, pp. 483–501 (page 49).

[118] Chunming Tang and Yuenai Chen. "Efficient Non-Interactive Verifiable Outsourced Computation for Arbitrary Functions." In: *IACR Cryptol. ePrint Arch.* 2014 (2014), p. 439 (page 49).

[119] Rosario Gennaro and Daniel Wichs. "Fully homomorphic message authenticators". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2013, pp. 301–320 (page 49).

[120] Shweta Agrawal and Dan Boneh. "Homomorphic MACs: MAC-based integrity for network coding". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2009, pp. 292–305 (page 49).

[121] Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M Reischuk. "ADSNARK: nearly practical and privacy-preserving proofs on authenticated data". In: *2015 IEEE Symposium on Security and Privacy*. IEEE. 2015, pp. 271–286 (page 50).

[122] Dario Fiore and Rosario Gennaro. "Publicly verifiable delegation of large polynomials and matrix computations, with applications". In: *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012, pp. 501–512 (page 50).

[123] Liang Feng Zhang and Reihaneh Safavi-Naini. "Verifiable delegation of computations with storage-verification trade-off". In: *European symposium on research in computer security*. Springer. 2014, pp. 112–129 (page 50).

[124] Yihua Zhang and Marina Blanton. "Efficient secure and verifiable outsourcing of matrix multiplications". In: *International Conference on Information Security*. Springer. 2014, pp. 158–178 (page 50).

[125] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. "Verifiable delegation of computation over large datasets". In: *Annual Cryptology Conference*. Springer. 2011, pp. 111–131 (page 50).

[126] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. "Signatures of correct computation". In: *Theory of Cryptography Conference*. Springer. 2013, pp. 222–242 (page 50).

[127] Qingji Zheng, Shouhuai Xu, and Giuseppe Ateniese. "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data". In: *IEEE INFOCOM 2014-IEEE conference on computer communications*. IEEE. 2014, pp. 522–530 (page 50).

[128] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. "Flexible and efficient verifiable computation on encrypted data". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2021, pp. 528–558 (page 50).

[129] Patrick Struck, Lucas Schabhüser, Denise Demirel, and Johannes Buchmann. "Linearly homomorphic authenticated encryption with provable correctness and public verifiability". In: *International Conference on Codes, Cryptology, and Information Security*. Springer. 2017, pp. 142–160 (pages 59, 60, 82, 86, 88).

[130] D. Catalano, A. Marcedone, and O. Puglisi. "Authenticating Computation on Groups: New Homomorphic Primitives and Applications". In: *ASIACRYPT 2014*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 193–212 (pages 59, 60, 82, 87, 88).

[131] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database. 2010". In: *URL http://yann. lecun. com/exdb/mnist* 7 (2010), p. 23 (page 64).

[132] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy". In: *International conference on machine learning*. PMLR. 2016, pp. 201–210 (pages 65, 83, 98).

[133] 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 (page 65).

[134] Herve Chabanne, Roch Lescuyer, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. "Recognition Over Encrypted Faces: 4th International Conference, MSPN 2018, Paris, France". In: 2019 (page 65).

[135] F. Bourse, M. Minelli, M. Minihold, and P. Paillier. "Fast Homomorphic Evaluation of Deep Discretized Neural Networks". In: *Proceedings of CRYPTO 2018*. Springer, 2018 (page 65).

[136] 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 (page 65).

[137] Martin Zuber, Sergiu Carpov, and Renaud Sirdey. *Towards real-time hidden speaker recognition by means of fully homomorphic encryption*. Cryptology ePrint Archive, Report 2019/976. 2019 (page 65).

[138] F. Boemer, Y. Lao, and C. Wierzynski. "NGraph-HE: A Graph Compiler for Deep Learning on Homomorphically Encrypted Data". In: *CoRR* (2018) (page 65).

[139] F. Boemer, A. Costache, R. Cammarota, and C. Wierzynski. "NGraph-HE2: A High-Throughput Framework for Neural Network Inference on Encrypted Data". In: *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. WAHC'19. 2019, pp. 45–56 (page 65).

[140] A. Brutzkus, O. Oren Elisha, and R. Gilad-Bachrach. "Low Latency Privacy Preserving Inference". In: *Proceedings of the 36th International Conference on MachineLearning, Long Beach, California, PMLR 97*. 2019 (page 65).

[141] A. Sanyal, M. Kusner, A. Gascón, and V. Kanade. "TAPAS: Tricks to Accelerate (encrypted) Prediction As a Service". In: *ICML*. June 2018 (pages 65, 83, 98).

[142] E. Hesamifard, H. Takabi, and M. Ghasemi. "Deep Neural Networks Classification over Encrypted Data". In: *ACM CODASPY*. 2019, pp. 97–108 (pages 65, 83, 98).

[143] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and Li Fei-Fei. "Faster CryptoNets: Leveraging Sparsity for Real-World Encrypted Inference". In: *CoRR* (2018) (page 65).

[144] Marshall Ball, Brent Carmer, Tal Malkin, Mike Rosulek, and Nichole Schimanski. *Garbled Neural Networks are Practical*. Cryptology ePrint Archive, Report 2019/338. 2019 (page 66).

[145] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. "DeepSecure: Scalable Provably-Secure Deep Learning". In: *CoRR* (2017) (page 66).

[146] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. "Safetynets: Verifiable execution of deep neural networks on an untrusted cloud". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4672–4681 (pages 66, 83, 99).

[147] Lingchen Zhao, Qian Wang, Cong Wang, Qi Li, Chao Shen, Xiaodong Lin, Shengshan Hu, and Minxin Du. "VeriML: Enabling Integrity Assurances and Fair Payments for Machine Learning as a Service". In: *arXiv preprint arXiv:1909.06961* (2019) (page 66).

[148] Hervé Chabanne, Julien Keuffer, and Refik Molva. "Embedded Proofs for Verifiable Neural Networks." In: *IACR Cryptol. ePrint Arch.* 2017 (2017), p. 1038 (page 66).

[149] Seunghwa Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. "vCNN: Verifiable Convolutional Neural Network." In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 584 (page 66).

[150] Julien Keuffer, Refik Molva, and Hervé Chabanne. "Efficient proof composition for verifiable computation". In: *European Symposium on Research in Computer Security*. Springer. 2018, pp. 152–171 (page 66).

[151] Jens Groth. "On the size of pairing-based non-interactive arguments". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2016, pp. 305–326 (page 66).

[152] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic methods for interactive proof systems". In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 859–868 (page 66).

[153] Edouard Dufour Sans, Romain Gay, and David Pointcheval. "Reading in the Dark: Classifying Encrypted Digits with Functional Encryption." In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 206 (pages 66, 77, 78).

[154] *Microsoft SEAL (release 3.0)*. http://sealcrypto.org. Microsoft Research, Redmond, WA. Oct. 2018 (pages 77, 108).

[155] Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Jeffrey Hoffstein, Kristin Lauter, Satya Lokam, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. *Security of Homomorphic Encryption*. Tech. rep. Redmond WA, USA: HomomorphicEncryption.org, July 2017 (page 78).

[156] Martin Zuber and Dario Fiore. *HAL: A Library for Homomorphic Authentication*. http://www.myurl.com. 2017 (pages 78, 108).

[157] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. 2017, pp. 1273–1282 (pages 81, 84).

[158] *GDPR 2018 reform of EU data protection rules*. European Commission. 2018. URL: https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf (visited on 06/17/2019) (page 82).

[159] Centers for Medicare & Medicaid Services. *The Health Insurance Portability and Accountability Act of 1996 (HIPAA)*. Online at http://www.cms.hhs.gov/hipaa/. 1996 (page 82).

[160] B. Hitaj, G. Ateniese, and F. Ferez-Cruz. *Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning*. 2017. arXiv: 1702.07464 [cs.CR] (pages 82, 84, 90, 96).

[161] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. "Exploiting unintended feature leakage in collaborative learning". In: *IEEE SP*. 2019, pp. 691–706 (page 82).

[162] M. Fredrikson, S. Jha, and T. Ristenpart. "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures". In: *ACM SIGSAC*. 2015, pp. 1322–1333 (pages 82, 84, 96).

[163] Le Trieu Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. *Privacy-Preserving Deep Learning via Additively Homomorphic Encryption*. Cryptology ePrint Archive, Report 2017/715. https://eprint.iacr.org/2017/715. 2017 (pages 82, 83).

[164] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. "Federated machine learning: Concept and applications". In: *ACM TIST* 10.2 (2019), pp. 1–19 (pages 82, 83).

[165] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. "Practical Secure Aggregation for Privacy-Preserving Machine Learning". In: *ACM SIGSAC*. 2017, pp. 1175–1191 (pages 82, 84).

[166] V. Mugunthan and A. Polychroniadou. "SMPAI: Secure Multi-Party Computation for Federated Learning". In: 2019 (pages 82, 84).

[167] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. "Verifynet: Secure and verifiable federated learning". In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 911–926 (pages 82, 83).

[168] P. Kairouz et al. "Advances and Open Problems in Federated Learning". arXiv preprint 1912.04977. 2019 (page 82).

[169] Flavio Bergamaschi, Shai Halevi, Tzipora T Halevi, and Hamish Hunt. "Homomorphic Training of 30,000 Logistic Regression Models". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2019, pp. 592–611 (page 83).

[170] S. Carpov, N. Gama, M. Georgieva, and J. R. Troncoso-Pastoriza. *Privacy-preserving semi-parallel logistic regression training with Fully Homomorphic Encryption*. Cryptology ePrint Archive, Report 2019/101. https://eprint.iacr.org/2019/101. 2019 (page 83).

[171] Qian Lou, Bo Feng, Geoffrey C Fox, and Lei Jiang. "Glyph: Fast and Accurately Training Deep Neural Networks on Encrypted Data". In: *arXiv preprint arXiv:1911.07101* (2019) (pages 83, 99, 103).

[172] Wenting Zheng, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. "Helen: Maliciously secure coopetitive learning for linear models". In: *2019 IEEE SP)*. IEEE. 2019, pp. 724–738 (page 83).
[173] Junyi Li and Heng Huang. "Faster Secure Data Mining via Distributed Homomorphic Encryption". In: *ACM SIGKDD*. 2020, pp. 2706–2714 (page 83).
[174] Sinem Sav, Apostolos Pyrgelis, Juan R Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. "POSEIDON: Privacy-Preserving Federated Neural Network Learning". In: *arXiv preprint arXiv:2009.00349* (2020) (page 83).
[175] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning". In: *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*. 2020, pp. 493–506 (page 83).
[176] Florian Tramer and Dan Boneh. "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware". In: *arXiv preprint arXiv:1806.03287* (2018) (pages 83, 99).
[177] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. "Learning differentially private language models without losing accuracy". In: *arXiv preprint arXiv:1710.06963* (2017) (page 84).
[178] Robin C Geyer, Tassilo Klein, and Moin Nabi. "Differentially private federated learning: A client level perspective". In: *arXiv preprint arXiv:1712.07557* (2017) (page 84).
[179] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. "Distributed learning without distress: Privacy-preserving empirical risk minimization". In: *NeurIPS* 31 (2018), pp. 6343–6354 (page 84).
[180] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. "cpsgd: Communication-efficient and differentially-private distributed sgd". In: *NeurIPS* 31 (2018), pp. 7564–7575 (page 84).
[181] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. "Membership inference attacks against machine learning models". In: *IEEE SP*. IEEE. 2017, pp. 3–18 (pages 84, 96).
[182] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. "White-box vs black-box: Bayes optimal strategies for membership inference". In: *ICML*. 2019, pp. 5558–5567 (pages 84, 96).
[183] S. Caldas, S.M. Karthik Duddu, P. Wu, T. Li, J. Konečný, H.B. McMahan, V. Smith, and A. Talwalkar. "LEAF: A Benchmark for Federated Settings". arXiv preprint 1812.01097. 2019 (page 91).
[184] Xiaoxin Yin, Jiawei Han, Jiong Yang, and Philip S Yu. "Efficient classification across multiple database relations: A crossmine approach". In: *IEEE Transactions on Knowledge and Data Engineering* 18.6 (2006), pp. 770–783 (page 97).
[185] Ludmila I Kuncheva and Juan J Rodriguez. "Classifier ensembles with a random linear oracle". In: *IEEE Transactions on Knowledge and Data Engineering* 19.4 (2007), pp. 500–508 (page 97).
[186] Elena Baralis, Silvia Chiusano, and Paolo Garza. "A lazy approach to associative classification". In: *IEEE Transactions on Knowledge and Data Engineering* 20.2 (2007), pp. 156–171 (page 97).

[187] Tatiana Tommasi and Barbara Caputo. "The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories". In: *BMVC*. CONF. 2009 (page 98).

[188] Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. "Safety in numbers: Learning categories from few examples with multi model knowledge transfer". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 3081–3088 (page 98).

[189] Hua Wang, Feiping Nie, Heng Huang, and Chris Ding. "Dyadic transfer learning for cross-domain image classification". In: *2011 International conference on computer vision*. IEEE. 2011, pp. 551–556 (page 98).

[190] Yusuf Aytar and Andrew Zisserman. "Tabula rasa: Model transfer for object category detection". In: *2011 international conference on computer vision*. IEEE. 2011, pp. 2252–2259 (page 98).

[191] Luo Jie, Tatiana Tommasi, and Barbara Caputo. "Multiclass transfer learning from unconstrained priors". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 1863–1870 (page 98).

[192] Yin Zhu, Yuqiang Chen, Zhongqi Lu, Sinno Jialin Pan, Gui-Rong Xue, Yong Yu, and Qiang Yang. "Heterogeneous transfer learning for image classification". In: *Twenty-Fifth AAAI Conference on Artificial Intelligence*. 2011 (page 98).

[193] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. "From n to n+ 1: Multiclass transfer incremental learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3358–3365 (page 98).

[194] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. "Transfer feature learning with joint distribution adaptation". In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 2200–2207 (page 98).

[195] Novi Patricia and Barbara Caputo. "Learning to learn, from transfer learning to domain adaptation: A unifying perspective". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1442–1449 (page 98).

[196] Marcus Rohrbach, Sandra Ebert, and Bernt Schiele. "Transfer learning in a transductive setting". In: *Advances in neural information processing systems* 26 (2013), pp. 46–54 (page 98).

[197] Nitish Srivastava and Ruslan Salakhutdinov. "Discriminative Transfer Learning with Tree-based Priors." In: *NIPS*. Vol. 3. 4. Citeseer. 2013, p. 8 (page 98).

[198] Wei Wang, Hao Wang, Chen Zhang, and Fanjiang Xu. "Transfer feature representation via multiple kernel learning". In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015 (page 98).

[199] Xianbin Cao, Zhong Wang, Pingkun Yan, and Xuelong Li. "Transfer learning for pedestrian detection". In: *Neurocomputing* 100 (2013), pp. 51–57 (page 98).

[200] Bernardino Romera-Paredes, Min SH Aung, Massimiliano Pontil, Nadia Bianchi-Berthouze, Amanda C de C Williams, and Paul Watson. "Transfer learning to account for idiosyncrasy in face and body expressions". In: *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. IEEE. 2013, pp. 1–6 (page 98).

[201] Jin Gao, Haibin Ling, Weiming Hu, and Junliang Xing. "Transfer learning based visual tracking with gaussian processes regression". In: *European conference on computer vision*. Springer. 2014, pp. 188–203 (page 98).

[202] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. "Transfer learning through greedy subset selection". In: *International Conference on Image Analysis and Processing*. Springer. 2015, pp. 3–14 (page 98).

[203] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. "Scalable greedy algorithms for transfer learning". In: *Computer Vision and Image Understanding* 156 (2017), pp. 174–185 (page 98).

[204] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014) (pages 98, 105).

[205] Xingquan Zhu and Xindong Wu. "Class noise handling for effective cost-sensitive learning by cost-guided iterative classification filtering". In: *IEEE Transactions on Knowledge and Data Engineering* 18.10 (2006), pp. 1435–1440 (page 98).

[206] Qiang Yang, Charles Ling, Xiaoyong Chai, and Rong Pan. "Test-cost sensitive classification on data with missing values". In: *IEEE Transactions on Knowledge and Data Engineering* 18.5 (2006), pp. 626–638 (page 98).

[207] Masayuki Tanaka. "Learnable image encryption". In: *2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*. IEEE. 2018, pp. 1–2 (page 98).

[208] Warit Sirichotedumrong, Takahiro Maekawa, Yuma Kinoshita, and Hitoshi Kiya. "Privacy-preserving deep neural networks with pixel-based image encryption considering data augmentation in the encrypted domain". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 674–678 (page 98).

[209] Romain Mormont, Pierre Geurts, and Raphaël Marée. "Comparison of Deep Transfer Learning Strategies for Digital Pathology". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 2262–2271 (page 103).

[210] Francois Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015 (page 109).