Sorbonne Université

EDITE - Ecole Doctorale Informatique, Télécommunications et Electronique

EURECOM

Privacy-preserving Biometric Systems with Advanced Cryptographic Techniques

Par Alberto Ibarrondo Luis

Thèse de doctorat
Dirigée par Melek Önen & Hervé Chabanne

Présentée et soutenue publiquement le 27 Mars 2023 à Sophia Antipolis

Composition du jury:

Aikaterini Mitrokotsa, Professor, University of St. Gallen	*Rapportrice*
Thomas Schneider, Professor, Technical University of Darmstadt	*Rapporteur*
Antonio Faonio, Maître de Conférences, EURECOM	*Éxaminateur*
Melek Önen, Maître de Conférences (HDR), EURECOM	*Directrice de thèse*
Hervé Chabanne, Professor, Telecom Paris	*Co-Directeur de thèse*

Abstract

While data becomes the world's most valuable resource, its processing places the privacy of data subjects in jeopardy. Dealing with highly sensitive data, identity management systems must provide adequate privacy protection as they leverage biometrics technology. Modern privacy-preserving computation techniques rely on advanced cryptographic primitives including multi-party computation (MPC), homomorphic encryption (HE) or a combination of both. Wielding these primitives, as well as Functional Encryption (FE), this thesis tackles the design and implementation of practical privacy-preserving biometric systems, from the feature extraction to the matching with enrolled users.

This work is consecrated to the design of secure biometric solutions for multiple scenarios, putting special care to balance accuracy and performance with the security guarantees, while improving upon existing works in the domain. We go beyond privacy preservation against semi-honest adversaries by also ensuring correctness facing malicious adversaries. Last but just as important, we address the leakage of biometric data when revealing the output, a privacy concern often overlooked in the literature. The main contributions of this thesis are:

- A new optimized face identification solution built on FE-based private inner product matching with countermeasures against input leakage.
- A novel efficient two-party computation protocol, Funshade, to preserve the privacy of biometric thresholded distance metric operations.
- An innovative method to perform privacy-preserving biometric identification based on the notion of group testing named Grote.
- A new distributed decryption protocol with collaborative masking addressing the leakage for biometric systems, dubbed Colmade, and grounded on the Brakerski-Fan-Vercauteren scheme.
- An honest majority three-party computation protocol to perform maliciously secure inference of Binarized Neural Networks, Banners, relevant for biometric feature extraction.
- A HE library written in Python named Pyfhel, offering a high-level abstraction including low-level functionalities, with applications in teaching.

Keywords

Privacy, Secure computation, Biometrics, Face Identification, Multi-party Computation, Homomorphic Encryption, Functional Encryption, Scalar Product.

Résumé

Alors que les données deviennent la ressource la plus précieuse au monde, leur traitement met en péril la vie privée des personnes concernées. Traitant des données très sensibles, les systèmes de gestion de l'identité biométriques doivent assurer une protection adéquate. Les techniques modernes de calcul sécurisé reposent sur le calcul multipartite (MPC), le chiffrement homomorphe (HE) ou une combinaison des deux. En utilisant ces primitives, ainsi que le chiffrement fonctionnel (FE), cette thèse aborde la conception et la mise en œuvre de systèmes biométriques préservant la confidentialité, couvrant de l'extraction des caractéristiques biométriques à la reconnaissance des utilisateurs enregistrés.

Ce travail est consacré à la conception de solutions biométriques sécurisées pour de multiples scénarios pratiques, en mettant un soin particulier à équilibrer la précision et la performance avec la sécurité, tout en améliorant les travaux existants dans le domaine. Nous allons au-delà de la préservation de la confidentialité contre les adversaires semi-honnêtes en garantissant également la correction face aux adversaires malveillants. Enfin, nous abordons la fuite des données biométriques lors de la révélation du résultat, un problème de confidentialité souvent négligé dans la littérature. Les contributions de cette thèse sont:

- Une nouvelle solution d'identification des visages basée sur la FE pour des produits scalaires, avec des contre-mesures contre les fuites d'entrée.
- Un nouveau protocole de calcul à deux parties, FUNSHADE, préservant la confidentialité des opérations biométriques de calcul de distance avec seuil.
- Une méthode innovante d'identification biométrique préservant la confidentialité nommée GROTE, basée sur la notion de pooling.
- Un nouveau protocole de déchiffrement homomorphe distribué, COLMADE, avec masquage collaboratif abordant la fuite des systèmes biométriques.
- Un protocole de calcul tripartite à majorité honnête pour l'inférence avec des réseaux neuronaux binarisés, appelé BANNERS, sécurisé contre des adversaires malicieux.
- Une libraire Python nommée PYFHEL, offrant une abstraction de haut niveau des FHE avec une portée pédagogique.

Mots-clés

Confidentialité, Systèmes Biométriques, Identification Faciale, Calcul Multiparti, Chiffrement Homomorphe, Chiffrement Fonctionnel, Produit Scalaire.

Contents

Chapter 1

Introduction

Data could be labelled as the 21st century oil. There are numerous modern applications fueled by data, ranging from Data Analytics & Machine Learning to Biometrics to name a few, whose impact in society is undeniable. Indeed modern data analytics & Machine Learning (ML) have disrupted many market sectors, ranging from the entertainment industry and manufacturing (e.g., content prediction algorithms, automatic fault detection), to more sensitive areas like healthcare or the public administration (e.g., early-stage cancer detection, fraud prosecution).

Yet, the tremendous potential of data manipulation is coupled with high risks. Data misuse and theft, specially when dealing with personal data, are ever-present concerns that can be mitigated by resorting to privacy policies and techniques (as covered in present-day data protection legislations such as GDPR in Europe or HIPAA for medical records in United States).

These risks are exacerbated on certain applications. Biometric systems must rely on secure hardware or trusted parties to hold the personal data vital for their recognition models, and all the biometric data manipulation must follow strict security rules. Hospitals and health specialists are deprived of the advantages of training and using models with a high volume of data from patients, which has proven to be very effective at training accurate prediction models tackling complex problems, e.g., genome-wide association studies or image-based early cancer detection [182, 169]. Banks are limited to the locally available data to prevent fraud and prosecute tax evasion. Child Exploitative Imagery detection models [238] need training data that is in itself illegal to possess.

Mishandling privacy-sensitive data not only imposes concrete harms on the affected users, but also threatens the adoption of these new technological innovations. Today, industry best practices require that service providers protect personal data in-transit and at-rest using encryption. However, in applications where computation is to be performed on the data, this data must be decrypted before being used, requiring the service provider to have access to the keying material. This exposes the data to multiple threats, including abuse by actors with malicious intent.

Under the field of advanced cryptography, several privacy preserving technologies aim to deal with these issues. *Fully Homomorphic Encryption* (FHE) [109] is a costly encryption scheme that supports certain operations between ciphertexts (typically addition and multiplication), yielding the results of these operations when decrypting. Secure Multi-party Computation (MPC) covers a series of techniques (garbled circuits [244], secret sharing [220]) that split data across multiple distinct parties, so that each individual party remains ignorant of the global values, and collaborate to jointly compute a given function. *Functional Encryption* [34] is a public-key encryption scheme that supports evaluation of arbitrary functions when decrypting the ciphertexts, where the decryption key holds the information about the function to be computed, and the original data can only be retrieved with the original encryption key. FHE and MPC can also coexist in Multi-party Homomorphic Encryption [186] (MHE), where distributed versions of the FHE protocols are carried out by a number of collaborating parties.

Centering our attention in the field of biometrics, the use of biometric solutions for identification and authentication is becoming increasingly prevalent in a wide range of applications such as access control, financial transactions [177], and even voting systems [243]. Biometric data (e.g., fingerprint, face, iris) are unique to each individual and can be easily collected and verified, making them a convenient and secure form of identification. However, the collection, storage and usage of biometric data also raises significant privacy concerns. Biometric data is sensitive information that can be used to identify and track individuals, and if it falls into the wrong hands, it can be misused for nefarious purposes, such as identity theft or surveillance. Moreover, once compromised, it cannot be changed like a password, making it a permanent vulnerability. Therefore, it is crucial to develop privacy-preserving techniques that can protect the privacy of biometric data while maintaining the accuracy and usability of the underlying system. Such techniques can include the use of FHE, MPC or FE to perform computations on the biometric data in a secure and privacy-preserving manner. By applying these techniques, we can ensure that the biometric data is not revealed in the clear, and only authorized parties can access and use it.

The principal objective of this thesis is to develop and implement privacy-preserving techniques for biometric systems, from the extraction of biometric features to the identification and authentication of individuals. We make use of modern cryptographic techniques to design secure protocols for a wide range of scenarios, including malicious adversaries. We also analyze the shortcomings of their security guarantees leading up to practical attacks based on the revealed outputs, and we address with suitable countermeasures. Last but not least, we implement evaluate the performance of our solutions, demonstrating their applicability to real use-cases while maintaining high accuracy and usability.

1.1 Main Contributions

This thesis constitutes a comprehensive study of privacy-preserving biometric systems, and it presents a set of novel contributions that address different aspects on the road to build secure and practical biometric solutions. These contributions follow the natural structure of biometric systems, from the biometric features extraction of individuals to the verification of their identity. The main contributions of this thesis are summarized as follows, in the order they appear in the thesis:

I. The design and implementation of PYFHEL, a Python wrapper for the Microsoft SEAL [219] library, extendable to other C++ libraries, offering *(i)* one-click install, including the underlying back-end libraries, *(ii)* a high-level Python-first abstraction layer that makes working with FHE significantly easier, including *(iii)* high-level APIs for low-level functionalities not generally exposed. We show the usability of PYFHEL both to explore and to teach FHE.

II. A novel protocol for secure inference of binarized neural networks based on replicated secret sharing dubbed BANNERS. This protocol *(i)* guarantees security with abort against one malicious adversary in a 3-party setting, *(ii)* has a performance equivalent to existing state-of-the-art semi-honest protocols.

III. A new face identification solution built on FE-based private inner product matching. This solution *(i)* optimizes the online latency for the same security guarantees by switching functionalities of encryption and key generation FE algorithms, *(ii)* conducts a thorough security analysis of the inner product input leakage including countermeasures to thwart attacks based on it, and *(iii)* is tested and validated in a face matching scenario, attesting its applicability practical one-use identification use-cases.

IV. A novel two-party computation protocol named FUNSHADE to perform privacy - preserving distance metric computations with a subsequent comparison to θ, built upon a combination of advanced Secret Sharing [193] and Functional Secret Sharing [38]. This protocol *(i)* requires just one round of communication in the online phase, lowering the communication costs with respect to existing two-round protocols [213, 38], *(ii)* sends only two ring elements in the online phase, reducing the communication size of previous solutions by a factor of $2l$ (for vectors of length l), *(iii)* features 100% correctness in the comparison result, and *(iv)* is implemented and open-sourced in a standalone Python library with efficient C++ primitives.

V. An innovative method to perform privacy-preserving $1 : K$ biometric identification based on the notion of group testing named GROTE. This method *(i)* replaces the K element-wise comparisons by group testing to reduce the number of such costly, non-linear operations in the encrypted computation, *(ii)* is instantiated and tested on FHE with the CKKS scheme, showing that it *(iii)* incurs a small impact in the accuracy of the system while speeding up its execution 1.5 times.

VI. A new decryption protocol with collaborative masking based on the multiparty variant [186] of the Brakerski-Fan-Vercauteren (BFV) [100] Homomorphic Encryption scheme named COLMADE. This protocol *(i)* performs a decryption in a distributed pool of users, employing them to mask masks a fragment of the ciphertext during decryption while remaining agnostic of the full computation, *(ii)* guarantees the privacy of all but one bit of the disclosed output in diverse threat models; *(iii)* is used to construct an auditable privacy-preserving biometric identification system; and *(iv)* is implemented, thoroughly tested and open-sourced.

Most of these contributions have been presented in conferences dedicated to the domain of privacy-preserving computation and published in their proceedings. We list below the main publications stemming from this thesis, in the order of their publication:

[132] *Banners: Binarized neural networks with replicated secret sharing.* **Alberto Ibarrondo**, Hervé Chabanne and Melek Önen. Presented at IH&MMSec2021.

[133] *Practical Privacy-Preserving Face Identification based on Function-Hiding Functional Encryption.* **Alberto Ibarrondo**, Hervé Chabanne and Melek Önen. Presented at CANS2021.

[136] *Pyfhel: Python for homomorphic encryption libraries*, **Alberto Ibarrondo** and Alexander Viand. Presented at WAHC2021.

[130] *Colmade: Collaborative Masking in Auditable Decryption for BFV-based Homomorphic Encryption.* **Alberto Ibarrondo**, Hervé Chabanne, Vincent Despiegel and Melek Önen. Presented at IH&MMSec2022.

[131] *Grote: Group Testing for Privacy-Preserving Face Identification.* **Alberto Ibarrondo**, Hervé Chabanne and Melek Önen. Accepted at CODASPY2023.

[134] *Funshade: Functional Secret Sharing for Two-Party Secure Thresholded Distance Evaluation.* **Alberto Ibarrondo**, Hervé Chabanne and Melek Önen. Under submission.

1.2 Overview

This thesis is organized as follows:

- Chapter 2 describes biometric systems separating the feature extraction from the verification, and outlines the requirements to impose in privacy-preserving biometric solutions while highlighting existing challenges.
- Chapter 3 introduces the cryptographic techniques used in the thesis and proposes a novel library to enhance usability of homomorphic encryption.
- Chapter 4 focuses on the protection of the feature extractor. Chapter 5 deals with the protection of biometric data and securing biometric verification.
- Chapter 6 is devoted to studying the privacy leaks when revealing the output and applying suitable mitigation strategies.
- Finally, Chapter 7 summarizes all the contributions of this thesis and discusses future research directions.

1.3 Notation

Scalars, vectors and polynomials. We use regular letters to denote scalars (e.g., r, θ) and bold letters for vectors of scalars (e.g., $\boldsymbol{x}, \boldsymbol{\theta}$). $\boldsymbol{x}^{(i)} = \boldsymbol{x}[i]$ both denote the ith element of vector $\boldsymbol{x}$. For convenience we omit the $^{(i)}$ superscripts in lengthy element-wise additions of the form $\Sigma[\boldsymbol{a}^{(i)} + \boldsymbol{b}^{(i)} + \ldots]$. We write $\boldsymbol{ab} = \boldsymbol{c}$ to denote the element-wise multiplication of two vectors where $\boldsymbol{c}^{(i)} = \boldsymbol{a}^{(i)}\boldsymbol{b}^{(i)}$, and $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^T\boldsymbol{b} = z$ to denote the inner (scalar) product between two vectors. In the domain of FHE we use regular letters also for polynomials, and boldface letters for vectors of polynomials. In this context, R_L expresses a polynomial ring with integer coefficients modulo L. $p[i]$ denotes the ith coefficient/element of a polynomial p.

Sampling and distributions. We note $\mathcal{U}_{[S]}$ to the uniform random distribution in the set S, and write $r \sim \mathcal{U}_{[S]}$ to sampling that distribution and assigning the sample to local variable r. $\mathcal{N}(\mu, \sigma)$ denotes a univariate gaussian distribution with mean μ and standard deviation σ. Given a sampling of an individual coefficient $p[j]$ from a distribution $\mathcal{D}$ ($p[j] \sim \mathcal{D}$), we denote the sampling of a polynomial p over a ring R_L as $p \leftarrow \mathcal{D}_{[R_L]}$.

Operations. We denote $[\cdot]_q$ the reduction modulo q, and $\lfloor\cdot\rfloor$, $\lfloor\cdot\rceil$, $\lceil\cdot\rceil$ the rounding to the previous, nearest and next integer respectively. When applied to polynomials or vectors, these reductions are performed coefficient/element-wise. For a polynomial a, we write its infinity norm as $\|a\|$. For an input integer $x \in \mathbb{Z}$ we use $sign(x) = x/|x|$, and define $sgb(x)$ as the sign bit such that:

$$sgb(x) = \begin{cases} 1 & \text{if } x < 0 \\ 0 & \text{if } x \geq 0 \end{cases}$$

We employ $(\cdot)_?$ to denote the Boolean evaluation of the expression inside the brackets, e.g., $(3 > 2)_? = 1$. Alternatively, we employ $\mathtt{1}_{x\in A}$ to denote the indicator function (e.g., $\mathtt{1}_{x>5} = 1 \;\Leftrightarrow\; x > 5$):

$$\mathtt{1}_{x\in A} \equiv \mathtt{1}_A(x) \triangleq \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A, \end{cases}$$

As a special case of indicator function, the unit step function is defined as $(x \geq 0)_? = \mathtt{1}_{x\in\mathbb{R}^*_+} = \mathtt{1}_{x\geqslant 0}$.

Twos-complement integer encoding. In contexts where numerical precision is defined in $\mathbb{Z}_M$ for $M = 2^m$, we implicitly consider a twos-complement encoding to map between signed and unsigned m-bit integers, a bijective mapping between $[-2^{m-1}, 2^{m-1} - 1]$ and $[0, 2^m - 1]$ by applying $\bmod 2^m$, where the interval of negative values $[-2^{m-1}, -1]$ is mapped to the upper half of the unsigned interval $[2^{m-1}, 2^m - 1]$. As such, the unit step function for m-bit integers corresponds to $\mathtt{1}_{x\in\mathbb{Z}_{2^m_+}} = \mathtt{1}_{0\leqslant x\leqslant 2^{m-1}-1}$.

Parties and MPC. We reserve the notation P_{descr} to indicate a party/entity in our scenario with a certain description (e.g., P_{setup} for the party in charge of the setup, more about this in Section 3.1). We P_a :SEND $r \Rightarrow \mathsf{P}_b$ for party a sending value r to party b. We use $\leftarrow$ for assignment to local variables, e.g. $\mathsf{P}_A : h \leftarrow 4$ instructs party A to set the value of h to 4. We leave the notation of the different sharing schemes for Section 3.2 (A brief summary can be found in Table 3.2).

Chapter 2

On Challenges in Privacy-Preserving Biometrics

As the main focus of this thesis, we start-off with an introduction to the field of biometrics, then cover the requirements for building privacy-preserving biometric systems,to close up with a discussion on the challenges we address throughout this thesis.

2.1 A Gentle Introduction to Biometric Systems

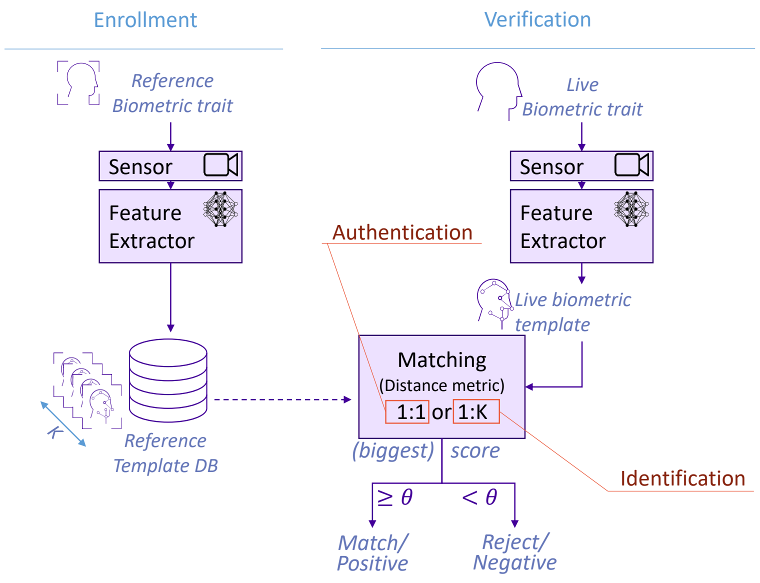


Figure 2.1: Diagram of a generic biometric system

Biometric systems are pattern recognition systems which establish the authenticity of a specific physiological or behavioral user's characteristic, relying on "who/what you are" to identify you. These characteristics, broadly named biometric traits, are scanned and compressed into succinct representations called **biometric templates**. Biometric recognition systems compare these templates to establish and verify the identity of users. The most commonly used biometric traits are face, iris and fingerprint [214]. Biometrics are broadly used in modern identification systems such as personal (mobile and laptop) authentication, identification for public administration, access control in restricted facilities, or border control/passenger identification in the travel industry to name a few.

In essence, biometric systems present two phases, depicted in Figure 2.1:

- The *enrollment phase* involves registering users by collecting their biometric templates and storing them in a database to serve as reference.
- The *verification phase* captures a live biometric template from a user seeking to authenticate/identify himself and matches this template with the reference templates stored in the database.

As shown in Figure 2.1, both phases share the template extraction process, which consists of a sensor (e.g., camera) capturing a biometric trait (e.g., taking a picture) and running it through a feature extractor. The feature extractor can be seen as a black box in charge of mapping the digitalized biometric traits into 1D vectors of fixed length (example of such length can be found in Table 2.1), the biometric templates. Modern feature extractors are based on Deep Learning models from the Computer Vision domain [214, 89, 90, 7] (see section 2.1.4 for an outline of these models). For a chosen distance metric, these models are trained to minimize the distance among templates belonging to the same identity, while maximizing the distance with respect to templates from all other identities.

Table 2.1: Typical template sizes for biometric applications

Biometric Trait	Typical template
Iris [86, 122]	640-1024 bits
Fingerprint [247]	40-100 minutae, 4 integer each
Face [89]	128|256|512 normalized floating point

Given two biometric templates obtained from the same feature extractor, the **matching** consists on evaluating the distance between them using f_{dist}, the metric adopted in the feature extractor, to assess the similarity between pairs of templates. Immediately after, the score is compared to a predefined threshold θ (Section 2.1.1 will describe how this threshold is set). The two most common metrics employed by biometric systems are [214]:

- The *hamming distance* $\mathrm{HD}(\boldsymbol{x}_b, \boldsymbol{y}_b) = \sum \mathbf{x}_b[i] \oplus \mathbf{y}_b[i]$ between two binary vectors $\boldsymbol{x}_b$ and $\boldsymbol{y}_b$, where $\oplus$ is the XOR operation. HD is commonly applied to fingerprint and iris biometrics.
- The *normalized scalar/inner product (a.k.a. cosine similarity)* ($\mathrm{SP}(\mathbf{x}, \mathbf{y}) = \overline{\mathbf{x}} \cdot \overline{\mathbf{y}} = \sum \overline{\mathbf{x}}[i] \cdot \overline{\mathbf{y}}[i]$, where $\overline{\mathbf{x}} = \mathbf{x}/\left\|\mathbf{x}\right\|$ is the L2 normalized template vector). SP is commonly used in face biometrics.

In the enrollment (a.k.a. setup) phase, the users being enrolled provide their biometric templates, which are stored in a database to be used as references in the verification phase. Then, during the verification phase the freshly extracted ("live") template $\boldsymbol{x}$ from a user seeking to verify his identity is matched to the reference templates $\boldsymbol{Y}$ in the database for an established metric. The matching process is generalized in Equation 2.1:

$$o = f_{verif}(f_{dist}, \boldsymbol{x}, \boldsymbol{Y}, \theta) \tag{2.1}$$

The matching/similarity score is calculated as the pair-wise distance $f_{dist}(\boldsymbol{x}, \boldsymbol{y})$ between the live template and each of the reference templates $\boldsymbol{y} \equiv \boldsymbol{Y}^{(k)}\ \forall k \in \{1, \ldots, K\}$, and the system compares these scores with the threshold θ to yield a positive result (a *match*) if at least one score is above the threshold and a negative result (a *reject*) otherwise (other types of outputs are discussed in Section 2.1.3). Depending on the number of templates in the enrollment database, we can have two scenarios:

- *Authentication*, a $1:1$ matching between the live template and a single reference template $\boldsymbol{y}$. It yields a positive result if the matching score is higher than a given threshold θ, negative otherwise. In a nutshell, it answers "Are you who you claim to be?".

- *Identification*, a $1:K$ matching between the live template and K stored templates belonging to up to K registered users. Requiring K distinct matchings, an identification request returns a *match* if the said score is above the threshold θ, and a negative result if not. Optionally, the ID of the reference template with highest matching score can be returned in case of a positive result. In short, it provides an answer to "Are you in the list of enrolled users?", or alternatively to "Who are you?".

2.1.1 Accuracy of Biometric Systems

Given the matching score between a pair of templates, the decision of rejecting or accepting that score as a match depends on the comparison to the threshold θ. Evidently, the choice of the value of θ directly impacts the chances of the biometric system to correctly identify previously enrolled users while rejecting non-registered users. θ is fixed as part of the system design. A high value of θ will impose a strong requirement in the matching score, meaning that registered users will have a harder time to be accepted, while non-registered users (a.k.a. *impostors*) will be rejected with higher probability. On the other hand, a low value of θ will make it easier for validly enrolled users to be accepted[1], while also increasing the chances of impostors being accepted.

The selection of θ is based on the formulation of biometric matching as a binary classification problem: dichotomizing the matching scores into "match" or "reject" groups [31]. Much like in many other practical binary classification problems, the two groups are not symmetric, and rather than overall accuracy, the relative proportion of different types of errors is of interest. For example, in medical testing, detecting a disease when it is not

[1]Accidentally, this can lead to an increase in the robustness of the system to variations in the biometric trait, such as lighting conditions, pose, partial cover, etc

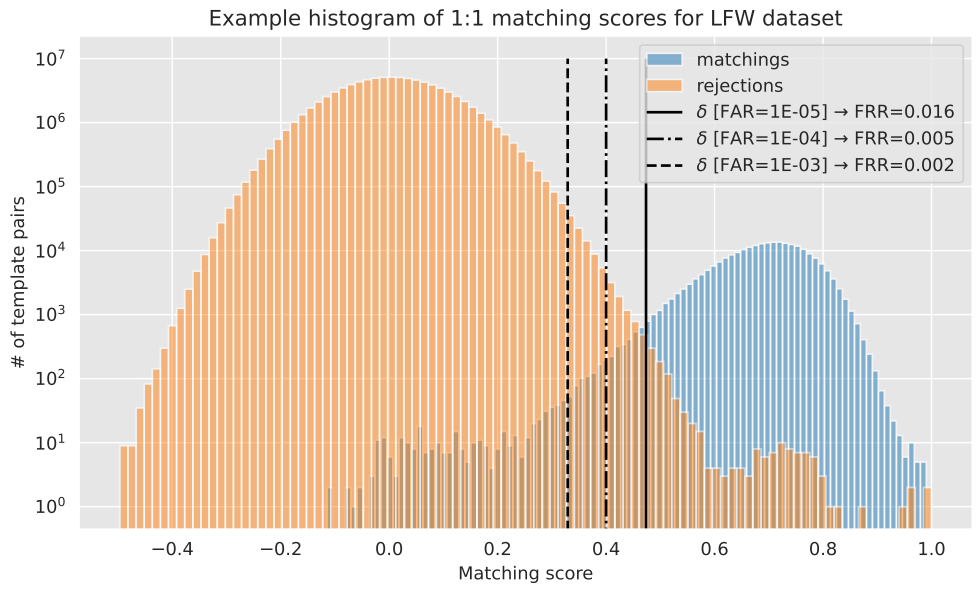


Figure 2.2: Histogram of matching scores for all possible template pairs in the Labelled Faces in the Wild (LFW [127]) dataset, using an Arcface-based feature extractor [89]. The vertical lines represent the values for threshold θ to achieve the fixed FAR rates of $\{10^-5, 10^-4, 10^-3\}$.

present (*a false positive*) is considered differently from not detecting a disease when it is present (*a false negative*).

In the domain of biometrics, we make use of dedicated datasets of biometric traits to calibrate the system (set θ) and assess its overall precision, composed of samples from many identities (ideally with abundant diversity in gender/age/race) with one or several samples per identity. This dataset is similar to the "test" dataset in Deep Learning, hinting that it must not overlap with the "training" dataset used to train the feature extractor model, as it would lead to a lack of generalization in the system (*overfitting*). One of the most popular datasets for testing/benchmarking biometric systems, is the *Labeled Faces in the Wild (LFW)* dataset [127], comprised of 13.233 images of 5.749 people collected from the web, which we use in our experiments.

Armed with such a dataset, we can measure the accuracy of a biometric system by running a verification experiment with pairs of templates[2], where one template acts as a reference and the other as the live template transforming the matching score into a binary decision (match/reject) and contrasting it with the ground truth (genuine user/impostor). With the matching scores of all the template pairs at hand, we are able to draw a plot like that of Figure 2.2. The accuracy of a biometric system is given by the *false acceptance rate* (FAR), the probability of a biometric system to accept an impostor, and the *false rejection rate* (FRR), the probability of a biometric system to reject a genuinely valid user. The FAR and FRR are defined as follows:

$$\begin{aligned} \text{FAR} &= \frac{\text{number of impostors accepted}}{\text{number of users accepted}} \\ \text{FRR} &= \frac{\text{number of genuine users rejected}}{\text{number of users rejected}} \end{aligned} \tag{2.2}$$

The FAR and FRR are inversely related, since increasing the threshold θ will decrease the FAR and increase the FRR. Typically, biometric systems are designed to yield a fixed value of FAR (e.g., 10^-3), setting up θ accordingly and obtaining a corresponding FRR. Figure 2.2 depicts an example of this binary classification task, including the FAR-FRR trade-off for fixed FAR values of 10^{-3}, 10^{-4} and 10^{-5}. In short, it is at this point that we set θ to achieve the desired balance between the FAR and the FRR of our biometric system for a target "test" dataset.

To condense the FAR-FRR balance into a single metric, one can use the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve, a curve defining the probability that a randomly chosen genuine user will have a higher matching score than a randomly chosen impostor (refer to section 5.2 of [31] for an in-depth description). Put more simply, the ROC is the probability of correctly identifying a genuine user (True Acceptance Rate or $TAR = 1 - FRR$) for each FAR rate, and the AUC is the area under this curve (its integral). The higher the AUC, the better the biometric system. As an alternative, the *equal error rate* (EER), defined as the point where the FAR and FRR are equal, can also be used as a single metric to evaluate the accuracy of a biometric system, although it is not as widely used as the AUC. That being said, modern biometric systems are designed to yield an AUC of 0.99 or higher, hence biometric benchmarks often rely on the FRR for fixed FAR to compare different systems [120].

2.1.2 A closer look at Face Identification

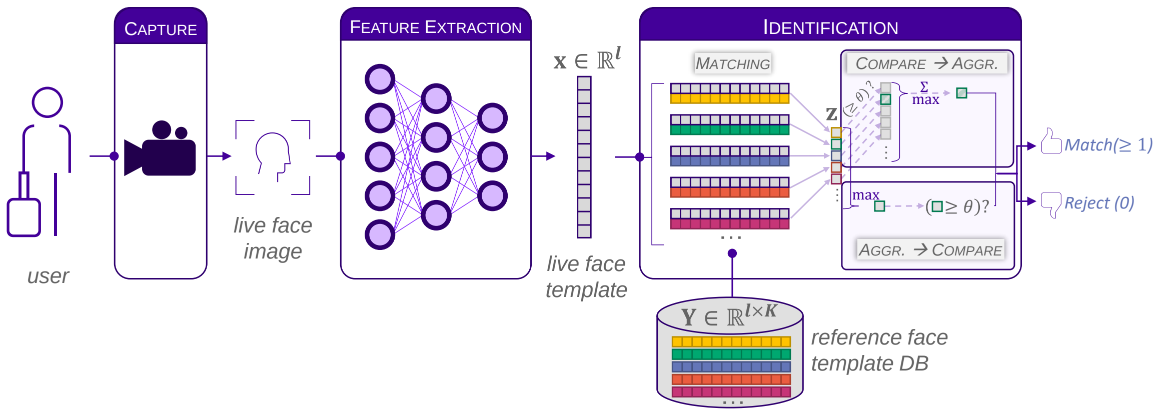


Figure 2.3: Verification phase in face identification systems.

[2]For small biometric datasets you can exhaustively run verification with all possible pairs of templates, and we do just that in all our experiments with LFW. However, larger datasets require a more efficient approach, leading to an representative selection of template pairs for the experiment.

Due to the relative importance of face biometrics in industry and in this thesis, we now take a closer look at the face identification problem, depicting its matching phase in Figure 2.3.

Modern feature extractors for faces such as ArcFace-based Neural Networks [89] generate large templates ($l \in 128, 256, 512$) with a considerable floating point precision. The numerical precision of the templates can be a requirement to yield low error rates, which must be taken into account when translating these values into integers as part of the adaptation required for their use alongside most Privacy Enhancing Technologies.

The enrollment and feature extraction work just like in the generic case. For the matching step, we are given a normalized input live template $\mathbf{x} \in \mathbb{R}^{l}_{[-1,1]}$ with l elements and a database of reference templates with K identities $\mathbf{Y} \in \mathbb{R}^{l\times K}_{[-1,1]}$ to compute identification based on *cosine similarity*, as described in Equation 2.3. First we obtain the matching scores $\mathbf{z} \in \mathbb{R}^{l\times K}_{[-1,1]}$ by performing K scalar products, and then obtain k^* (the index of the identity yielding a positive matching) by resorting to a composition of element-wise comparisons to the threshold θ and arg max of an array.

$$\begin{gathered}\mathbf{z} \triangleq\ z[k] = \sum_{i=1}^{l} \mathbf{x}[i] \cdot \mathbf{Y}[i,k] \\ \text{Compare first: } \mathbf{z}_\theta = (\mathbf{z} \geq \theta)_? \ \rightarrow\ k^* = \arg\max_k \{\mathbf{z}_\theta[k]\} \\ \text{Max first: } k^*_{\mathbf{z}} = \arg\max_k \{\mathbf{z}[k]\} \ \rightarrow\ k^* = k^*_{\mathbf{z}} \cdot (\mathbf{z}[k^*_{\mathbf{z}}] \geq \theta)_?\end{gathered} \tag{2.3}$$

The comparison to θ and arg max operations can be swapped, as they incur in a similar computational overhead. In "Max first" K comparisons in $\mathbb{R}^l$ are needed to get the max of a 1D score vector of K values (one per identity in the database), and one last comparison to θ. Conversely, "Compare first" requires K elementwise comparisons to θ in $\mathbb{R}^l$, followed by the arg max in a binary vector with highly sparse values(since typically up to 1 element is 1 and the rest are 0).

2.1.3 Output of a biometric verification

From the descriptions above, we can see that the output o of a biometric verification can come in different shapes, depending on the application. In the following, we consider the most common types of outputs and formalize them:

- A *"match"/"reject" binary output* corresponds to an output $o \in \mathbb{Z}_2$ indicating the outcome of the verification process, with $o = 1$ for a match and $o = 0$ for a reject. This is the most common output type for biometric authentication, and it applies to both authentication (1 : 1, "are you who you say you are?") and identification (1 : K, "are you in the list of enrolled users?").

$$o_{binary} = f_{verif-binary}(f_{dist}, \boldsymbol{x}, \boldsymbol{Y}) = \Big\{(f_{dist}(\boldsymbol{x}, \boldsymbol{Y}^{(k)}) \geq \theta)_?\ \forall k\Big\} \tag{2.4}$$

- A *"similarity score" output* yields the matching score directly in the result $o \in \mathbb{Z}_L$. It applies to a $1:1$ authentication scenario, and it is extensible to $1:K$ with K independent outputs:

$$o_{score} = f_{verif-score}(f_{dist}, \boldsymbol{x}, \boldsymbol{Y}) = \left\{ f_{dist}(\boldsymbol{x}, \boldsymbol{Y}^{(k)})\ \forall k \right\} \tag{2.5}$$

- A *"similarity score of max" output* is an alternative of the above for $1:K$ identification where only the highest matching score is returned:

$$o_{max-score} = f_{verif-max-score}(f_{dist}, \boldsymbol{x}, \boldsymbol{Y}) = \max \left\{ f_{dist}(\boldsymbol{x}, \boldsymbol{Y}^{(k)})\ \forall k \right\} \tag{2.6}$$

- An *"ID index" output* results in the index $o \in \mathbb{Z}_{\log_2 K}$ of the positively matching identity (e.g., $k*$ in Equation 2.3) in a $1:K$ identification scenario. This is the most common output type for $1:K$ identification:

$$o_{index} = f_{verif-index}(f_{dist}, \boldsymbol{x}, \boldsymbol{Y}) = \left(\arg\max \left\{ f_{dist}(\boldsymbol{x}, \boldsymbol{Y}^{(k)})\ \forall k \right\} \geq \theta \right)_? \tag{2.7}$$

2.1.4 Examining the feature extractor: Convolutional Neural Networks

As introduced in the previous section, the feature extractor is the component that transforms the input biometric data into a feature vector all across the system. As such, the feature extractor is at the core of a biometric system's precision, in charge of maximizing the distance between the feature vectors of different users, while minimizing the distance between the feature vectors of the same user.

Riding the wave of Computer Vision models based on Deep Learning (DL), modern feature extractors are nowadays Convolutional Neural Networks (CNN), a family of Neural Networks (NN) specialized in dealing with images. Much like all Machine Learning, these mathematical models go through two distinct phases: *Training*, where the model is adapted using vast amounts of data, and *Inference*, where the model is used to make predictions on new data. CNNs, and NNs more in general, fall into the category of *supervised learning*, where the data used to train the model is labeled. E.g.: a neural network trained to recognize handwritten digits requires the training dataset to contain both images of the handwriting samples and the corresponding actual value (from 0 to 9).

For biometrics, the *training phase* consists of solving a carefully crafted optimization problem by leveraging on massive amounts of labeled biometric images. The cost/loss function to minimize is designed with the goal of discerning identities (e.g., sampling template triplets consisting of two matching templates and a non-matching template [126]; shaping the template space to be a sphere surface [89] and pulling together same-identity samples while pushing away the rest). Later on, the available data is iterated multiple times in a computationally expensive process named *backpropagation* to steadily adapt the parameters and the architecture of the DL model. Training these CNN models is a gargantuan challenge of its own, and as such it is left out of the scope of this thesis, where we concentrate in its direct application as part of biometric systems.

Broadly speaking, there are two separate sections in a CNN model tailored for classification, portrayed in Figure 2.4:

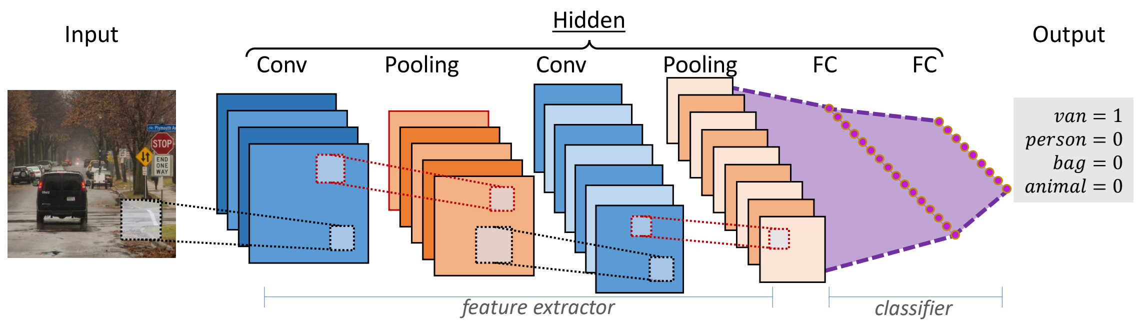


Figure 2.4: Example of a Deep Convolutional Neural Network

- **Feature Extractor**: extracts abstract information (*features*, e.g., detecting edges, shapes, gradients in a picture) from the input image that will help classify it. It outputs a vector of features (the *template* in biometric systems).
- **Classifier**: categorizes a feature vector into a set of classes (in the case of biometric systems, the set of identities), e.g., combine the detected edges/shapes/gradients to predict if the image contains a human face. This is much more personalized.

In the context of biometrics, the classifier obtained in the training phase is dropped, and the feature extractor is frozen and deployed all across biometric systems to perform inference on previously unseen biometric data on a continuous basis, as the system is used to authenticate/identify users.

Generically speaking, a CNN with L layers is composed of:

1. An **input layer**, the tensor of input data $\mathbf{A}_{in}$
2. $L-1$ **hidden layers**, mathematical transformations applied somewhat sequentially.
3. An **output layer**, the tensor of output data $\mathbf{A}_{out}$.

We denote the output of layer i as a tensor $\mathbf{A}^{[i]}$, with $\mathbf{A}^{[\mathbf{0}]} = \mathbf{A}_{in}$,and $\mathbf{A}^{[L]} = \mathbf{A}_{out}$. Tensors can have different sizes and even different number of dimensions.

Selecting a CNN *architecture* involves choosing the number, types, order, and size of layers. These choices are crucial for the performance of the CNN, hence a substantial amount of the research effort in the field of Deep Learning is devoted to the design and refinement of these architectures to solve specific problems. A comprehensive review of the state-of-the-art architectures for face and fingerprint recognition can be found in [157] and [82]. Nevertheless, we now review the most relevant layers used in CNNs.

2.1.4.1 Fully Connected Layer (FC)

Also known as Dense Layer, it is composed of N parallel elements a.k.a. *neurons*, performing the $\mathbb{R}^M \to \mathbb{R}^N$ transformation presented in Figure 2.5. For the ith layer with N output neurons and M input neurons (output of previous layer) we define:

$\mathbf{a^{[i]}} = \left[a_1^{[i]} \dots a_k^{[i]} \dots a_N^{[i]}\right]^T$ as output; $\mathbf{b^{[i]}} = \left[b_1^{[i]} \dots b_k^{[i]} \dots b_N^{[i]}\right]^T$ as bias;
$\mathbf{z^{[i]}} = \left[z_1^{[i]} \dots z_k^{[i]} \dots z_N^{[i]}\right]^T$ as linear output;

$$\mathbf{W^{[i]}} = \left[\mathbf{w_1^{[i]}} \quad \dots \quad \mathbf{w_k^{[i]}} \quad \dots \quad \mathbf{w_N^{[i]}}\right]^T = \begin{bmatrix} w_1^{[i]}[1] & \dots & w_N^{[i]}[1] \\ \vdots & \ddots & \vdots \\ w_1^{[i]}[M] & \dots & w_N^{[i]}[M] \end{bmatrix}^T, \text{ weight matrix.}$$

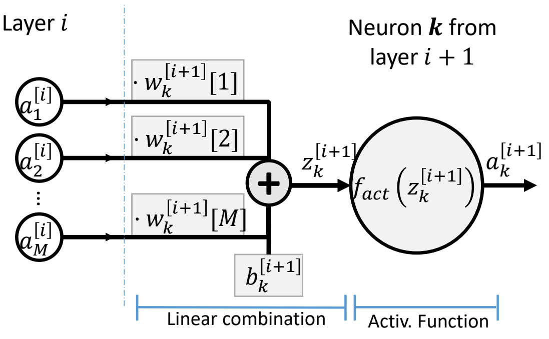


Figure 2.5: Fully Connected layer followed by Activation for neuron k

Neuron k performs a linear combination of the output of the previous layer $\mathbf{a^{[i-1]}}$ multiplied by the weight vector $\mathbf{w_k^{[i]}}$ and shifted with a bias scalar $b_k^{[i]}$, obtaining the linear combination $z_k^{[i]}$:

$$\begin{aligned} z_k^{[i]} = \left(\sum_{l=1}^{M} w_k^{[i]}[l] \cdot a_l^{[i-1]}\right) + b_k^{[i]} &= \left(\mathbf{w_k^{[i]}}\right)^T * \mathbf{a^{[i-1]}} + b_k^{[i]} \\ \mathbf{z^{[i]}} &= \left(\mathbf{W^{[i]}}\right)^T * \mathbf{a^{[i-1]}} + \mathbf{b^{[i]}} \end{aligned} \tag{2.8}$$

2.1.4.2 Activation Function

Activation functions are the major source of non-linearity in CNNs. They are performed element-wise (thus easily vectorized), and generally located after the linear combination in FC layers and after convolution in Convolutional layers (Equation 2.9).

$$a_k^{[i]} = f_{act}\left(z_k^{[i]}\right) \tag{2.9}$$

The absence of activation function ($a_k^{[i]} = z_k^{[i]}$), is also referred to as "Linear activation" or $f_{act}(x) = x$. Four major activation functions can be found in the literature:

• ***Sigmoid (σ)*** The classical activation function. It transforms negative values into the interval $(0, 0.5]$ and positive values into $[0.5, 1)$.

$$Sigmoid(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.10}$$

• ***Rectifier Linear Unit (ReLU)*** is one of the most widely used activation functions for CNN. It outputs 0 for negative values and z for positive values, thus accumulating successive activated neurons ($z > 0$) while disconnecting the inactivate ones ($z \leqslant 0$). Several variants such as Leaky ReLU (LReLU) or Parametrized ReLU (PReLU) [123] try to avoid dead neurons (neurons that don't activate for any input data).

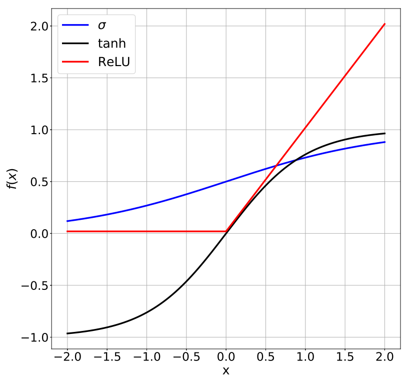


Figure 2.6: Activation functions

$$ReLU(z) = z^+ = max(0, z) \tag{2.11}$$

• ***Hyperbolic Tangent (tanh)*** is a shifted alternative of sigmoid to $(-1, 1)$. It transforms negative values into the interval $(-1, 0]$ and positive values into $[0, 1)$.

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.12}$$

• ***Softmax*** (Eq. 2.13) is used at the output layer of NNs only because its neurons across the whole layer sum to 1, allowing them to be considered as probabilities (in the interval [0,1]). This is very useful for classification, where you obtain the probabilities to belong to each class (a metric of confidence in the results).

$$Softmax(z_k) = \frac{e^{z_k}}{\sum_{k=1}^{N} e^{z_k}} \tag{2.13}$$

2.1.4.3 Convolutional Layers (Conv)

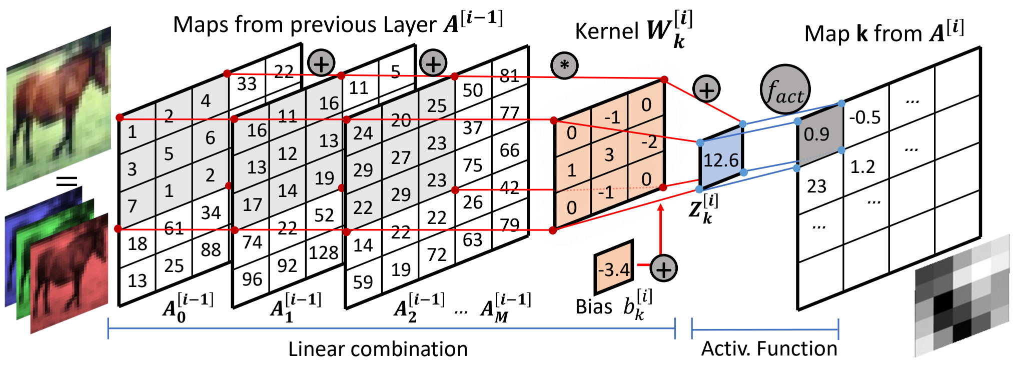


Figure 2.7: Convolutional Layer, naïve implementation

Convolutional layers (Figure 2.7) constitute an essential improvement for image recognition and classification using NNs. The linear transformation involved is *spatial convolution*

[162], where a 2D $F \times F$ kernel (*filter*) is multiplied element-wise to the 2D input image in patches at steps of a defined number of pixels (*strides*), added up and then shifted by a bias. For input data with several channels (*maps*, e.g., RGB counts as 3 channels), the filter is applied to the same patch of each map and then added up into a single value of the output image (cumulative sum across maps). A map in convolution layers is the equivalent of a neuron in Fully Connected layers. For the ith layer with N maps we define:

$\mathbf{A_k^{[i]}} = \begin{bmatrix} a_k^{[i]}[1,1] & \dots & a_k^{[i]}[1,W] \\ \vdots & \ddots & \vdots \\ a_k^{[i]}[H,1] & \dots & a_k^{[i]}[H,W] \end{bmatrix}^T$ as the $W \times H$ kth output map (typ. W=H);

$\mathbf{Z_k^{[i]}} = \begin{bmatrix} z_k^{[i]}[1,1] & \dots & z_k^{[i]}[1,W] \\ \vdots & \ddots & \vdots \\ z_k^{[i]}[H,1] & \dots & z_k^{[i]}[H,W] \end{bmatrix}^T$ as the $W \times H$ kth linear output;

$\mathbf{b^{[i]}} = \begin{bmatrix} b_1^{[i]} & \dots & b_k^{[i]} & \dots & b_N^{[i]} \end{bmatrix}^T$ as the bias;

$\mathbf{W_k^{[i]}} = \begin{bmatrix} w_k^{[i]}[1,1] & \dots & w_k^{[i]}[1,F] \\ \vdots & \ddots & \vdots \\ w_k^{[i]}[F,1] & \dots & w_k^{[i]}[F,F] \end{bmatrix}^T$ as the $F \times F$ filter/kernel generating map k.

The linear operation in the convolution layer, transforming M maps from layer $i-1$ ($\forall m \in 1 \leqslant m \leqslant M$) into the map k from layer i ($\forall k \in 1 \leqslant k \leqslant N$) is expressed as:

$$\mathbf{Z_k^{[i]}} = \left(\sum_{m=1}^{M} \mathbf{A_m^{[i-1]}} \oplus \mathbf{W^{[i]}}_k \right) + b_k^{[i]} \tag{2.14}$$

The convolutional layer can be seen as a matrix to matrix multiplication using Toeplitz matrices that smartly replicate the filter weights, effectively transforming it into a FC layer.

2.1.4.4 Pooling Layer

This layer, depicted in Figure 2.8, reduces the input size by using a packing function. Most commonly used functions are `max` and `mean`. Similarly to convolutional layers, pooling layers apply their packing function to patches of the image separated by defined strides.

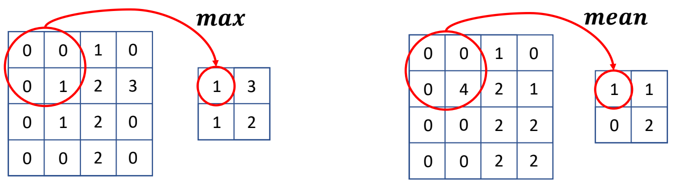


Figure 2.8: 2 : 1 Max and Mean packing example in a Pooling layer

In practice, pooling layers are set after one or several layers and used to reduce dimensionality of the inputs, as depicted in Figure 2.8

2.1.4.5 Batch Normalization

Batch Normalization [139] (BN) is a layer that is trained over batches (sets) of input data. The BN layer 'normalizes' the input corresponding to one data point with respect to the inputs from the entire batch: subtracting mean of the batch and dividing by its standard deviation. It is generally applied before activation functions to compact the values in a small interval around $x = 0$, obtaining a distribution that makes smarter use of non-linearities. BN helps preventing the model training from getting stuck in saturated modes (e.g., dead neurons), also helping to handle gradient explosion.

In a Batch Normalization layer there are two trainable parameters γ and β, optimized using backpropagation. β is a shifting parameter, while γ is a scaling parameter. Mean μ_B and variance σ_B^2 of each batch are calculated and stored. To avoid zero division, BN also includes a very small constant $\epsilon \approx 10^{-9}$. Once the network has been trained, the values of the mean and variance are fixed during inference. The BN formula is:

$$a^{[i]} = BN_{\gamma,\beta,\mu_T,\sigma_T^2}(x) = \gamma * \frac{a^{[i-1]} - \mu_T}{\sqrt{\sigma_T^2 + \epsilon}} + \beta \tag{2.15}$$

During inference, the operations of a BN layer located right before or after a linear layer (FC, Conv) can be absorbed[3] by that same linear layer [135], thus avoiding the computation of BN almost entirely.

2.1.4.6 Other CNN operations

- **Residual block** is an aggregation of layers where the input is added unaltered to the output of the transformations, thus allowing the transformations just to learn incremental (*residual*) modifications to the input. An example is shown in Figure 2.9:

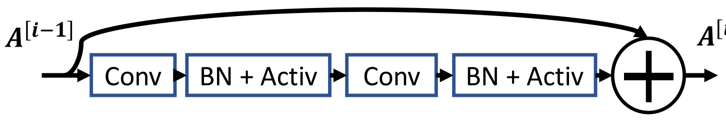


Figure 2.9: Residual Block, atomic element of ResNet [248]

- **Data preprocessing** comprises many techniques relying in prior knowledge of the input dataset, where you apply handpicked functions to the data in order to extract meaningful features later injected into the CNN.

2.1.5 Face identification instantiated as a FC layer

Armed with the taxonomy of CNN layers, we can reformulate the face identification operations as a FC layer and an activation function, where the live biometric template $\mathbf{x}$ corresponds to the input of the FC layer $\mathbf{a}$, the reference template database $\mathbf{Y}$ acts as

[3]Also referred to as *BN folding*, *BN fusing* or *BN reparameterization/reformulation*

the matrix of weights $\mathbf{W}$, the threshold θ as the bias b, and the activation function is $sign(z) = (z \geq 0)_?$[4]. Starting with Equation 2.3, we obtain the following equation:

$$\begin{gathered}\mathbf{x} \equiv \mathbf{a}_{in}; \qquad \mathbf{Y} \equiv \mathbf{W}; \qquad \theta \equiv b; \qquad f_{act}(z^{[k]}) = (z^{[k]} \geq 0)_? \\ k^* = \arg\max_k \{((\mathbf{x}^T * \mathbf{Y}) \geq \theta)_?\} \longrightarrow a_{out} = \arg\max_k \{f_{act}(\mathbf{a}_{in}^T * \mathbf{W} - \theta)\}\end{gathered} \tag{2.16}$$

While it might seem trivial at first glance, this transformation lets us use tools exposing FC and $f_{act}(z) = sgn(z)$ to instantiate the face identification process, as it is be the case with certain secure computation techniques.

2.2 Requirements for Privacy-Preserving Biometric Systems

The information regarding a user's biometric template or its identity is the target of a plethora of attacks on biometric systems [222, 6, 190], such as the recovery of the live biometric template and the reference templates, the traceability of a user across multiple services, or the deanonymization of the user corresponding to a given template. Given that biometric traits can be neither easily modified (despite the popularity of plastic surgery) nor re-issued, designing biometric systems that thwart or even impede these attacks is of paramount importance. Indeed, regulation bodies such as the European Union's General Data Protection Regulation (GDPR) [75] and the California Consumer Privacy Act (CCPA) [108] have recently introduced new privacy regulations that require the protection of biometric data. In this context, privacy-preserving biometric systems have emerged as a promising solution to address the privacy concerns of biometric systems, while seeking to maintaining their high accuracy and performance.

In line with all this, we are ready to list the requirements we expect our privacy-preserving biometric solutions to satisfy:

(R1) - Privacy: The system must preserve the privacy of:

- The biometric templates of the enrolled users (*Reference DB privacy*). The reference template database constitutes the single most important element to protect in biometric systems, since it contains the biometric information of all the enrolled users, and thus its disclosure to an adversary could allow it to impersonate enrolled users.
- The biometric templates of the users being verified (*Live template privacy*). While not as impactful to the overall biometric system, the live template is also worthy of protection, since it contains the biometric information of the users being verified. Its disclosure could lead to severe concerns to the privacy of these users: e.g., user tracking system-wise (or even user tracking across different systems), or the possibility of impersonating the user being verified.

[4]Note that this activation function can be written as $dReLU(z) = \partial\ \partial z(ReLU(z)) = sgn(z)$

- The parameters of the feature extractor used by the system (*Feature extractor privacy*). Modern feature extractors are very complex and costly to develop (computational resources, data required, technical expertise), and as such they are a valuable asset to protect, as an attempt to preserve the Intellectual Property rights held by its owner.

(R2) - Irreversibility: In order to guarantee that when a biometric storage database is attacked, attackers cannot recover the user's genuine private biometric information through the data stored in the database, these transformations must be irreversible. Thus, given a protected template, an entity holding the said template should not be able to recover the original biometric template unless it has access to the secret material.

(R3) - Unlinkability: Making users' real biological information unconnected from the outside. It is desirable to use systems employing altered or indirectly generated data for verification. Since the actual biometric traits are not connected to digital systems[5], there is a much lower probability that it will be compromised by corresponding attacks launched from the network. Hence, given a protected/secured template, an entity holding said template should not be able to link the encrypted biometric template of a user to his/her identity.

(R4) - Correctness: The system should be resilient to malicious tampering during the verification phase. It might be possible for an adversary to tamper with the computations of this phase, and thus it is important to understand if the system was subject of such an attack and provide countermeasures.

(R5) - Accuracy preservation: The protected system must be able to correctly verify valid users with a high probability (low FRR), while also rejecting impostors with a high probability (low FAR).

As a fundamental requirement of any biometric system, their privacy-preserving counterparts must also uphold this requirement. We expect the accuracy of the system to be reasonably preserved after adapting it with privacy-preserving solutions. This requirement arises from the fact that the privacy-preserving solutions we are considering are not perfect, and thus they might introduce calculation errors in the operations.

(R6) - Performance preservation: The secure version of the biometric system should not introduce an excessive overhead in terms of computational complexity or storage requirements. In other words, require an acceptably low verification latency while being able to handle a sufficiently large number of users.

In short, the system must be able to scale to the needs of the application it would be deployed in. This is of particular importance in the context of biometric systems, since they are often deployed in large-scale scenarios, such as airports, where they must be able to handle a large number of users in a short amount of time.

[5]That is, unless we place ourselves in some ultra-connected Metaverse-like scenario

2.3 Challenges

The main objective of this thesis is to study and apply suitable Privacy Enhancing Technologies (PETs, defined in Chapter 3) for the domain of biometric systems that address the requirements defined above. In line with this, we now outline the main challenges to face throughout this thesis,

(CH1): Deal with all the aspects that incur in accuracy losses, designing secure biometric systems preserve the accuracy (R5) of their non-secure counterparts. E.g., computation of non-linearities, encoding and fixed-point precision of the input data, etc.

(CH2): Address the tradeoffs between privacy (R1), accuracy (R5) and performance (R6). Find optimal Solutions for each scenario by leveraging off the available techniques to reach a compromise between computation costs, communication and accuracy.

(CH3): Going beyond protections against semi-honest adversaries by addressing malicious adversaries, guaranteeing not only privacy (R1) but also correctness (R4). We seek stronger security guarantees that are more suitable for deployment in real-world scenarios, dealing with solutions that entail higher complexity.

(CH4): Address the various template recovery attacks that arise from the information revealed during the execution and as an output of PETs, ensuring irreversibility (R2) and unlinkability (R3).

(CH5): Implement ready-to-use solutions in expressive programming languages to facilitate the adoption of the contributions of this thesis by the community.

2.3.1 On CH1: Accuracy Losses (R5)

The first challenge is a straightforward one: a privacy-preserving biometric system must keep the accuracy of the original biometric system to a high degree. The use of PETs often forces us to make approximations and transformations to the original operations while adapting the system to the constrains imposed by the chosen technique. These modifications have an unclear impact in the accuracy of the system. Therefore, it is vital to understand how these transformations affect the accuracy of the system.

While non-secure biometric systems typically use the full floating-point precision, PETs work in the integer domain (as we will see in Chapter 3). To bridge the two, we will rely on fixed-point encoding: upscaling floating-point inputs by a *scale* and rounding to the nearest integer. We need to study the impact of this fixed-point approximation on the accuracy (in the biometric sense) of the system.

Additionally, implementing the non-linearities of the biometric systems (e.g., *max* of matching scores, comparison to θ) with PETs draws forth many complications, as many PETs do not natively support linear non-linear functions at the same time. To overcome this, we are forced to either approximate the non-linearities with linear functions (requiring a high multiplication depth for polynomial approximations), or switch to different PETs designed for that purpose. We will study the impact of these approximations on the accuracy of the system.

2.3.2 On CH2: Balancing Privacy-Accuracy-Performance (R1)-(R5)-(R6)

Different PETs have different tradeoffs between privacy, accuracy and performance. For instance, *Multi-party Computation* (MPC) techniques [220, 244, 112] can offer strong simulation-based information-theoretic privacy guarantees and fairly accurate non-linear functions at the cost of a considerable amount of communication (both in terms of size and number of rounds). Conversely, *Fully Homomorphic Encryption* (FHE) [109] relies on computational hardness assumptions, only supports linear functions and polynomial evaluation natively, and entails a relatively high computational cost, but in exchange the operations can be performed by a single party incurring in low communication costs [6].

These aspects are to be considered when selecting the PETs to be used in a given scenario. Throughout this thesis we will study a wide variety of PETs, leading to contrasting trade-offs.

2.3.3 On CH3: Beyond Semi-Honest Adversaries (R4)

The main focus of this thesis is to study and apply PETs to the domain of biometrics. However, the biometric systems we will be dealing with may not only used in benign scenarios whereby all parties are assumed to follow the established protocol, but also in highly adversarial ones.

Dealing with malicious adversaries is a challenging task, as the adversary is not expected to deviate from the protocol specifications as it is the case with Semi-Honest adversaries. Instead, it is allowed to tamper the inputs, the outputs, or even the computations themselves. This is a very powerful capability, as it allows the adversary to completely subvert the biometric system, and thus it is important to design the system in a way that it is resilient to this behavior up to a certain degree.

2.3.4 On CH4: Addressing Leakage in the Output (R2)-(R3)

The output of a biometric system is usually a binary decision: either the user is verified (optionally outputting the nonzero ID of the user being verified) or not. Privacy-preserving biometric solutions are designed to disclose only this information (R2, R3). However, the revealed result of privacy-preserving solutions often entail much larger output spaces. Adversaries may leverage on this to extract information about the biometric inputs (R2), or to expose the true identity behind the inputs of the system (R3) while being compliant with the protocol.

This is a practical privacy concern often overlooked by the literature, and thus it is important to address it. Thus, studying the impact of the output leakage on the privacy of the system is of critical importance, and solutions to mitigate this issue are required.

[6] Accounting for the key distribution and the transmission of inputs and outputs

2.3.5 On CH5: Implementing Ready-to-Use Solutions

This thesis has a strong focus on the practical aspects of dealing with PETs. Consequently, we pursue implementations of all the proposed algorithms and protocols.

To promote the adoption of the contributions of this thesis by both the experts and the non-experts in the field, we aim to implement them in expressive programming languages of widespread adoption, favoring Python for high level interactions resembling as much as possible the scientific description of the algorithms/protocols, and resorting to other languages (e.g., C/C++) for efficient implementations of the cryptographic primitives.

2.4 Summary

This chapter covered a comprehensive description of biometric systems and their accuracy metrics, illustrating the case of face identification and diving into the feature extractor. Subsequently, we have inferred a set of requirements that privacy-preserving biometric systems should uphold: privacy, irreversibility, unlinkability, correctness, and accuracy & performance preservation. Lastly, we listed the five main challenges we are to face when building these systems: dealing with accuracy losses, balancing privacy with accuracy and performace, tackling malicious adversaries, addressing the leakage in the output and implementing ready-to-use solutions. We are now set to introduce the main privacy-preserving tools at our disposal (Chapter 3), to later apply them at several stages in biometric systems: the feature extraction (Chapter 4), the template verification (Chapter 5) and the output reveal (Chapter 6).

Chapter 3

Cryptography for Privacy-Preserving Operations

To undertake all the challenges and design biometric solutions fulfilling the requirements of Chapter 2, we have a set of privacy-preserving cryptographic tools at our disposal. We dedicate this chapter to comprehensively describe Multi-Party Computation (MPC), Fully Homomorphic Encryption (FHE) and Functional Encryption (FE), the main privacy-preserving techniques we are to employ in our solutions. We then discuss the strengths and weaknesses of each technique as well as available implementations, devoting the reminder of this chapter to our novel FHE library named Pyfhel.

3.1 Defining the Environment and the Adversaries

Prior to the analysis of the various cryptographic tools, we must define the context in which they operate. In particular, we set our sights to define the environment in which the interactions take place, the parties and their roles, and the adversaries that we are trying to protect against.

On the Environment. The environment defines the set of characteristics and assumptions that we make about the context in which the privacy-preserving protocols take place and where adversary operates. As part of this environment we designate a "party" to represent an actual real-world entity, consisting of a set of one or several Interactive Turing Machines [49] capable of executing Probabilistic Polynomial Time (PPT) algorithms that may use randomness to produce non-deterministic results[1].

In consonance with this, we assume the following:

- There exist *secure bidirectional communication channels* between each party and every other party (one could informally visualize it as a pre-established VPN) such that an eavesdropper can only observe the messages exchanged, but not modify or drop them.

[1]We borrow Definition 4 from [49], informally extending a classical Turing Machine with the ability to interact with other machines that may or may not belong to the same party. This can be seen as a generalization of the Turing Machine pairs defined in [115] for the case of multiple parties.

In practice, we instantiate this assumption by using standard cryptography (e.g., TLS) to secure the communication between the participants.

- These communication channels are *authenticated*, meaning that the sender of each message is verified. This can be achieved in practice by resorting to using digital signatures, a standard cryptographic primitive that allows the sender to sign a message with a private key, and the receiver to verify the signature with the corresponding public key.
- In protocols with more than two participants, there exists a *secure broadcast channel* for each participant to send a message to all the other participants at once. In practice, in the absence of a real broadcast channel we can achieve this using multiple secure and authenticated channels [161].
- Each party can access a *secure random number generator* (RNG) that produces random bits that are indistinguishable from random bits produced by a truly random source. In practice, this is achieved by employing cryptographically secure pseudo-random (deterministic) number generators (CS-PRNG, or PRNG to abbreviate).

A quick corollary of these assumptions is that the adversary is unable to tamper with the communication channels themselves (e.g., execute a man-in-the-middle attack).

On Roles and Parties. Our solutions are set a context of where several distinct entities, the **parties** $\{\mathsf{P}_A, \mathsf{P}_B, \dots\}$, take various **roles** to carry out their jobs as part of the system. In total, we distinguish up to five different roles in our privacy-preserving biometric systems and model them as:

- R_{setup}: The setup role is responsible for generating the preprocessing material during the offline phase, and distribute it to the parties involved in the online phase. As explained above, the setup party must be trusted due to its critical involvement in the enrollment phase.
- $\mathsf{R}_{in_{bmx}}, \mathsf{R}_{in_{bmy}}$: These are the owners/holders of the input vectors respectively, to be shared with the computing parties at the beginning of the online phase. By convention, and following Equation 2.3, we employ bmx for the live template and bmy/bmY for the reference template(s).
- $\mathsf{R}_0, \mathsf{R}_1, \dots$: These are the computing parties in charge of performing the secure computation. FHE and FE are designed for a single computing party, whereas MPC covers protocols defined for two or more computing parties.
- R_{out}: This is the party that will receive the result of the secure biometric verification process.

We write $\mathsf{P}_{ID} \supseteq \mathsf{R}_{descr}$ to denote the party P_{ID} identified by a tag ID that takes on the role R_{descr}, and generalize any party taking that role as P_{descr}. A role can be performed by more than one party at the same time, e.g., $\mathsf{P}_{setup_0}, \mathsf{P}_{setup_1}$ jointly perform the role

R_{setup}. Oppositely, multiple roles can coexist in a single party, e.g., two parties $(\mathsf{P}_A, \mathsf{P}_B)$ in charge of sharing the inputs and performing the computation can be denoted jointly as $(\mathsf{P}_A, \mathsf{P}_B) \supseteq (\mathsf{R}_{in}, \mathsf{R}_{\{0,1\}})$, or separately as $\mathsf{P}_A \equiv \mathsf{P}_{in_{\boldsymbol{x}}} \equiv \mathsf{P}_0$ and $\mathsf{P}_B \equiv \mathsf{P}_{in_{\boldsymbol{y}}} \equiv \mathsf{P}_1$. Lastly, we refer to the generic computing role as R_j, and the generic computing party as P_j.

On the Adversary. In cryptography, an adversary is a malevolent entity whose aim is to prevent the users of a cryptosystem from achieving their goals (fulfilling (R1)-(R6) in our case). An adversary's efforts might take the form of attempting to discover secret data, corrupting some of the data in the system, or impersonating a message sender or receiver. There are several types of adversaries depending on what capabilities they are presumed to have. Adversaries can be categorized as:

- *Computationally bounded* or *unbounded*, according to the computational power of the adversary (time and memory). Fencing off an adversary with infinite computing power means he will be unable to break a given requirement (e.g., privacy of a ciphertext) no matter how much time and computation he uses. Applied to privacy (R1), this translates into *perfect hiding* (e.g., even if he were to test all the possible private values leading to a ciphertext, he cannot decide which one is more probable as all of them are equally possible).

  Conversely, facing an adversary with limited computational power assumes that your opponent will eventually give up. Applied to privacy, it leads to *computational hiding* (e.g., he cannot figure out what you were hiding with a finite amount of computational power and time, except with negligible probability).

- *Static* or *adaptive* [50], depending on whether the adversary is constrained to choose its set of corruptions in advance (static) or allowed to adapt his choices of who/what to corrupt during the course of a protocol. This thesis restricts to static adversaries, as they are easier to analyze while still providing a level of security that is sufficient for our purposes.

- *Semi-honest* or *malicious*, depending on whether the adversary is allowed to interact with the system. A **semi-honest** adversary (a.k.a. Honest-but-Curious, passive) defines a type of adversary that will follow the instructions of a given protocol while trying to extract as much information as possible from the process. It requires passive security to overcome, making sure that data remains private throughout the protocol execution, but without the need to verify the result of the operations. In contrast, a **malicious** adversary (a.k.a. active) can deviate arbitrarily from the requested computation, forcing the verification of every step in a protocol to ensure correctness. This is a more generic security requirement, and it is often coupled with higher complexity.

  Since semi-honest adversaries require stronger assumptions (e.g., the adversary does not deviate from the protocol), protocols providing security against them are often seen as a stepping stone towards security against malicious adversaries.

3.2 Multi-Party Computation (MPC)

Secure multi-party computation (or MPC) [112, 58, 244] allows two or more parties to compute any function on private inputs without revealing anything but the output of the function. As long as their security assumptions are met, MPC protocols executed by N computing parties should be equivalent in terms of security to an idealization where all N parties send their inputs to a Trusted Third Party (TTP), the TTP computes all the protocol steps locally, and parties receive their corresponding output from the TTP. In this idealization, the TTP is assumed to be a trusted entity that cannot be corrupted by any adversary. In practice, the TTP is replaced by the set of N computing parties that are assumed to be mutually distrusting.

Importantly, those parties cannot learn any information beyond what is already known and permissible by the computation, meaning that MPC (much like FHE or FE) do not provide any kind of protection against function inversion (e.g., inferring the inputs of a function given its outputs). We deal with this fundamental limitation in Chapter 6.

The Pre-processing Model. Sometimes referred to as the offline-online paradigm, it is a common approach to outsourced secure computation and MPC in particular, where the protocol is split into two phases:

- **Pre-processing** (a.k.a. setup/offline) **phase**: the parties involved in the protocol perform the input-independent operations, that involve generation of key material, the generation of randomness with special properties that will be consumed in the online phase, and the encryption/sharing of data available ahead of time. It covers R_{setup} for every protocol, and R_{in_v} for system inputs v that are known ahead of time.
- **Evaluation** (a.k.a. online) **phase**: the parties involved in the protocol perform the input-dependent operations, including encryption/sharing of data available only at evaluation time, computation of the secure operations, and the decryption/reveal of the output of the protocol. It covers R_{in_x} for inputs known only at the beginning, R_j for the secure computations and R_{out} for the output of the protocol.

The pre-processing phase is employed to offload computation from the online phase, seeking to speed up the input-dependent operations in order to obtain solutions with low latency. In the context of biometric systems, it covers the entire enrollment phase including the protection (via encryption/sharing) of the reference biometric database. The pre-processing phase can be outsourced to a TTP, trusted by all other parties to not deviate from the protocol and to not leak any information. This is of crucial importance in the context of biometric systems, since leaking or maliciously modifying the reference template database automatically breaks biometric-based verification.

On the number of parties and majorities in MPC. An honest majority comprises strictly less than half of the computing parties being corrupted by an adversary, whereas a dishonest majority involves at least half of the computing parties being potentially corrupted. Generally speaking, the complexity of MPC intensifies with the number of parties and with the proportion of dishonest parties. Typically the best setup for dishonest majorities is Two Party Computation (2PC), as there is only one other party to blame when detecting a failure. In contrast, 3PC is particularly beneficial for an honest majority setting, since in this setting each honest party can rely on the honesty of at least one other party to check the correctness of the results. By comparing the results of the other two parties for a given computation, an honest party in this setting can cheaply detect malicious behavior and abort the computation [14, 104].

On families of MPC techniques. Historically there are two main approaches to MPC. In *Garbled Circuits(GC)* a computing party named the *garbler* encrypts a Boolean circuit in the form of truth tables using keys from each of the parties and randomly permutes the rows. Later on, the *evaluator* collaborates to sequentially decrypt single rows of the logic gates' truth tables while the garbler remains oblivious to the information exchanged between them by using a primitive named Oblivious Transfer. The second approach, named *Secret Sharing(SS)*, splits each individual data element into N shares sending one share per party, so that less than t shares reveal nothing of the input and t or more shares reconstruct the original secret without error. This approach offers cheap addition operations on local shares and communication between parties to exchange shares without ever revealing more than $t-1$ shares to any computing party. There are multiple variants of secret sharing, ranging from Shamir's polynomial-based sharing [220], to arithmetic and boolean sharing [112].

This thesis makes extensive use of SS and its variants, leading us to describe them in detail below. We focus on the two-party ($N = 2$) and three-party ($N = 3$) scenarios (2PC and 3PC respectively), as they are the most relevant for our work due to their simplicity and efficiency. We follow the notation from Table 3.2, covering three important variants of SS with distinct properties:

- *Replicated Secret Sharing (RSS)*, protecting against one malicious corruption in a 3PC setting.
- Π *Secret Sharing (ΠSS)*, greatly reducing the communication cost of a scalar product with respect to SS in a 2PC setting.
- *Functional Secret Sharing (FSS)*, secret sharing a function description instead of a secret value, and offering a very competitive comparison protocol for a 2PC setting.

Additionally, we denote the next (respectively previous) party to party j in the set $\{\mathsf{P}_0, \mathsf{P}_1, \ldots, \mathsf{P}_{N-1}\}$ as P_{j+1} (respectively P_{j-1}).

Table 3.1: Notation for the various secret sharing schemes: SS (Secret Sharing), RSS (Replicated Secret Sharing), ΠSS (Π Secret Sharing) and FSS (Functional Secret Sharing).

Scheme	#PC	Type	Share Notation	Properties
SS	N	Arithmetic ($\mathbb{Z}_L$)	$\mathsf{P}_i : \langle x\rangle_i$	$\sum_{i=0}^{N-1}\langle x\rangle_i \mod L = x \quad \forall x\in\mathbb{Z}_L$
	N	Binary ($\mathbb{Z}_2$)	$\mathsf{P}_i : [x]_i$	$\bigoplus_{i=0}^{N-1}[x]_i \mod 2 = x \quad \forall x\in\mathbb{Z}_2$
RSS	3	Arithmetic ($\mathbb{Z}_L$)	$\mathsf{P}_i : \langle\!\langle x\rangle\!\rangle_i \equiv (\langle x\rangle_i, \langle x\rangle_{i+1})$	$\langle x\rangle_0+\langle x\rangle_1+\langle x\rangle_2 \mod L = x \quad \forall x\in\mathbb{Z}_L$
	3	Binary ($\mathbb{Z}_2$)	$\mathsf{P}_i : [\![x]\!]_i \equiv ([x]_i, [x]_{i+1})$	$[x]_0\oplus[x]_1\oplus[x]_2 \mod 2 = x \quad \forall x\in\mathbb{Z}_2$
ΠSS	2	Arithmetic ($\mathbb{Z}_L$)	$\mathsf{P}_i : \langle\!\vert x\vert\!\rangle_i \equiv (\Delta_x, \delta_{x_i})$	$\Delta_x+\delta_{x_0}+\delta_{x_1} = x \quad \forall x\in\mathbb{Z}_L$
FSS	2	$\mathbb{Z}_L\to\mathbb{Z}_L$	$\mathsf{P}_i : \langle f\rangle_i$	$\langle f\rangle_0(x)+\langle f\rangle_1(x) = f(x) \quad \forall x\in\mathbb{Z}_L$

3.2.1 Secret Sharing (SS)

We define secret sharing in rings $\mathbb{Z}_L$, where a secret value x is split into N random shares $x_j,\ j\in\{0,\dots,N-1\}$ such that $x=\Sigma_{j=0}^{N-1}x_j \mod L$. The shares are distributed to the computing parties $\mathsf{P}_0,\mathsf{P}_1,\dots,\mathsf{P}_{N-1}$ such that party P_j holds the share x_j. With this sharing scheme, parties can perform local addition/subtraction of shared values. Additionally, parties can resort to Beaver's multiplication triples [22] to perform multiplication at the cost of one round of communication. At the end of the computation, the shares of the result are sent to P_{out} to reveal this result. Equation 3.1 shows the addition and multiplication in a 2PC setting. Note that operations involving P_j apply to all parties $\mathsf{P}_j\forall j\in\{0,\dots,N-1\}$:

$$
\begin{array}{lll}
\texttt{SS.share}(x)\to\langle x\rangle & \mathsf{P}_{in_x}\text{:} & \langle x\rangle_1 \sim \mathcal{U}_{[\mathbb{Z}_L]} \\
 & & \langle x\rangle_0 \leftarrow x-\langle x\rangle_1 \\
 & & \textsc{Send}(\langle x\rangle_j) \Rightarrow \mathsf{P}_j \\[1em]
\texttt{SS.reveal}(\langle x\rangle)\to x & \mathsf{P}_j\text{:} & \textsc{Send}(\langle x\rangle_j) \Rightarrow \mathsf{P}_{out} \\
 & \mathsf{P}_{out}\text{:} & x=\langle x\rangle_0+\langle x\rangle_1 \\[1em]
\texttt{SS.add}(\langle x\rangle,\langle y\rangle)\to\langle x+y\rangle & \mathsf{P}_j\text{:} & \langle x+y\rangle_j=\langle x\rangle_j+\langle y\rangle_j \\[1em]
\texttt{SS.mult}(\langle x\rangle,\langle y\rangle)\to\langle x\cdot y\rangle & \mathsf{P}_{setup}\text{:} & \langle a\rangle,\langle b\rangle \sim \mathcal{U}_{[\mathbb{Z}_L^{2\times 2}]};\quad \langle c\rangle\leftarrow\langle a\cdot b\rangle \\
 & & \textsc{Send}(\langle a\rangle_j,\langle b\rangle_j,\langle c\rangle_j) \Rightarrow \mathsf{P}_j \\
 & \mathsf{P}_j\text{:} & \textsc{Send}(\langle x\rangle_j-\langle a\rangle_j,\langle y\rangle_j-\langle b\rangle_j) \Rightarrow \mathsf{P}_{1-j} \\
 & & \langle x\cdot y\rangle_j=\langle b\rangle_j(x-a)+\langle a\rangle_j(y-b) \\
 & & \qquad +\langle c\rangle_j+j\cdot(x-a)(y-b)
\end{array}
\tag{3.1}
$$

The adaptation to N parties is straightforward. In practice we work with $L=2^w$ for values of $w\in\{8,16,32,64\}$ to benefit from a considerable speed up when dealing with w-bit modular arithmetic thanks to native integer types present in modern computers. Of special interest for this thesis, computing a scalar product $SP(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{l} \boldsymbol{x}^{(i)} \cdot \boldsymbol{y}^{(i)}$ with SS requires sending N terms per multiplication and party (for a total of $N^2 l$ values sent across the network), followed by local addition of the results of each element-wise multiplication.

3.2.2 From SS to Replicated Secret Sharing (RSS) in 3PC

Shifting our attention to the the 3PC setting ($N = 3$), Secret Sharing consists of splitting a secret integer $x \in \mathbb{Z}_L$ into randomly selected shares so that $x = \langle x \rangle_0 + \langle x \rangle_1 + \langle x \rangle_2$ and then sending each share $\langle x \rangle_j$ to $\mathsf{P}_j \forall j \in \{0, 1, 2\}$, allowing local addition and multiplication with one round of communication. Comparatively, the Replicated Secret Sharing technique coined by [14, 104] builds upon SS, joining two SS shares into an RSS share[2]:

$$\begin{aligned} \langle\!\langle x \rangle\!\rangle &\equiv [(\langle x \rangle_0, \langle x \rangle_1),\ (\langle x \rangle_1, \langle x \rangle_2),\ (\langle x \rangle_2, \langle x \rangle_0)] \\ &\equiv [\langle\!\langle x \rangle\!\rangle_0, \langle\!\langle x \rangle\!\rangle_1, \langle\!\langle x \rangle\!\rangle_2] \end{aligned} \tag{3.2}$$

and sends each RSS share $\langle\!\langle x \rangle\!\rangle_j$ to the corresponding party P_j (`RSS.share`). Given that only $t = 2$ shares are needed to reconstruct x, RSS is also designated as 2-out-of-3 secret sharing. Addition (`RSS.add`) and multiplication (`RSS.mult`) are executed independently and simultaneously for each of the SS shares that compose a RSS share, following the protocols from Equation 3.1 for each SS share.

The advantage of RSS over SS is that, thanks to the redundancy, parties can exchange intermediate results in pairs and verify that they both obtained the same result, e.g., after executing `RSS.mult` as two instances of `SS.mult` in parallel, P_0 holding $\langle\!\langle z \rangle\!\rangle_0 \equiv (\langle z \rangle_0, \langle z \rangle_1)$ and P_1 holding $\langle\!\langle z \rangle\!\rangle_1 \equiv (\langle z \rangle_1, \langle z \rangle_2)$ exchange their local copy of $\langle z \rangle_1$, convincing each other of the honest behavior if their copies match and stopping (*aborting*) the protocol execution otherwise. As a sought-off property, RSS provides security with abort against one malicious corruption in a 3PC honest majority setting, at the cost of one additional round of communication per multiplication.

Below we describe the use of *correlated randomness* to share a secret without communication, as well as the instantiation of SS and RSS for integer ($L = 2^w$ for w bits) and binary shares ($L = 2$) that will be used in the thesis, following the first four rows of table 3.2.

3.2.2.1 Correlated randomness for cheap 3PC sharing

Following the techniques of [14], after a setup phase where common seeds are exchanged, all parties can locally compute, correlated randomness α_j with the property $\alpha_0 + \alpha_1 + \alpha_2 = 0$ and uniformly random in $\mathbb{Z}_K$ using synchronized PRNGs: It suffices for each pair of computing parties $(\mathsf{P}_0, \mathsf{P}_1), (\mathsf{P}_1, \mathsf{P}_2), (\mathsf{P}_2, \mathsf{P}_0)$ to initialize a PRNG with a random seed agreed upon by each pair $(s_{(0,1)}, s_{(1,2)}, s_{(2,0)})$, and then locally generate $\alpha_j \leftarrow PRNG(s_{(j,j+1)}) - PRNG(s_{(j,j-1)})$.

[2]Instead of defining $\langle\!\langle x \rangle\!\rangle_j = (\langle x \rangle_j, \langle x \rangle_{j+1})$ as in [240], it might be convenient to define $\langle\!\langle x \rangle\!\rangle_j = (\langle x \rangle_j, \langle x \rangle_j \pm \langle x \rangle_{j+1})$ as in the original paper [14]. We still resort to the first option for simplicity.

This randomness can be used by party P_j to secret share a value x without communication by defining $\langle x\rangle = \Big(\langle x\rangle_j = x + \alpha_j,\ \langle x\rangle_{j+1} = \alpha_{j+1},\ \langle x\rangle_{j-1} = \alpha_{j-1}\Big)$. `RSS.share` can similarly benefit from this technique.

3.2.2.2 Arithmetic sharing in 3PC

Also known as Integer/Additive sharing, sharing a single integer $x \in \mathbb{Z}_{2^l}$ requires first to split x into random arithmetic shares by using randomness $\langle r\rangle_j$ uniformly random $\in \mathbb{Z}_{2^l}$ so that $r = 0 = \langle r\rangle_0 + \langle r\rangle_1 + \langle r\rangle_2$. These values are used to conceal x: $\langle x\rangle_j = x - \langle r\rangle_j$, as it naturally holds that $x = \langle x\rangle_0 + \langle x\rangle_1 + \langle x\rangle_2$. Note that $\langle r\rangle_j = \langle x\rangle_{j+1} + \langle x\rangle_{j-1}$ also holds true.

Following an RSS scheme, each party P_j receives $\langle\!\langle x\rangle\!\rangle_j = (\langle x\rangle_j, \langle x\rangle_j + \langle x\rangle_{j+1}) = (\langle x\rangle_j, \langle r\rangle_{j-1})$ when sharing an integer secret. Addition of two integer shared secrets can be computed locally on the parties [14], and multiplication between two integer shared secrets requires one round of communication with 2 integers sent per party [14, 104].

3.2.2.3 Binary sharing in 3PC

Similarly, sharing a single bit $b \in \mathbb{Z}_2$ requires first to split b into random bit shares by using some correlated randomness $[s]_j$ uniformly random $\in \mathbb{Z}_2$ so that $s = 0 = [s]_0 \oplus [s]_1 \oplus [s]_2$. These random values are used to conceal b: $[b]_j = b \oplus [b]_j$, and it naturally holds that $b = [b]_0 \oplus [b]_1 \oplus [b]_2$. Note that the equation $[s]_j = [b]_{j+1} \oplus [b]_{j-1}$ also holds true. Following an RSS scheme, each party P_j receives $[\![b]\!]_j = ([b]_j, [b]_{j-1})$ when sharing a binary secret. Analogous to integer sharing, XOR of two binary shared secrets can be computed locally on the parties [14], whereas AND between two binary shared secrets requires one round of communication with 2 bits sent per party [14, 104].

3.2.2.4 Share conversion

While arithmetic sharing is the natural choice to perform additions and multiplications, binary sharing can be much more effective to deal with non-linear functions, such as the comparison of two integers (e.g., by extracting the sign of the difference in a binary circuit). Much like ABY [88] for 2PC, ABY^3 [185] provides all the necessary tools to convert between arithmetic and binary shares (as well as garbled circuits) in a 3PC setting, and we will use these tools in the thesis.

3.2.3 Π-Secret Sharing (ΠSS) for efficient 2PC scalar product

Originally inspired by ASTRA [57] in the 3PC scenario, ABY2.0 [193] introduces a novel way to perform additive secret sharing in 2PC[3], where a secret value x is split into three random shares $(\Delta_x, \delta_{x_0}, \delta_{x_1})$ such that $\Delta_x = x + \delta_{x_0} + \delta_{x_1} \mod L$. The δ-shares δ_{x_j} are distributed to each computing party P_j forming an arithmetic secret sharing $\langle\delta_x\rangle$ of $\delta_x = \delta_{x0} + \delta_{x1}$, while the Δ-share Δ_x is held by both parties at once. We name this sharing scheme as Π-secret sharing, due to the "horizontally" mutual Δ-share and the two "vertically" separated δ-shares, and denote the Π-sharing of value x as $\langle\![x]\!\rangle$. The Π-sharing scheme allows local addition/subtraction, and multiplication at the cost of one round of communication. The essential difference with respect to the SS scheme is that the δ-shares can be pre-computed, and thus carry extra correlation[4] that wasn't possible with standard SS. The main arithmetic operations in ΠSS are defined as in Equation 3.3.

Crucially, the online phase of the Π-sharing multiplication first computes a local arithmetic sharing of the result $\langle x \cdot y \rangle$, and then uses one round of communication to convert the result back into Π-shares. As promptly explained in [193], this moves the communication from the multiplication inputs (see `SS.mult` in Equation 3.1) to the multiplication outputs, which yields sizeable advantages in terms of communication size for operations such as the scalar product: computing a scalar product $SP(\boldsymbol{x}, \boldsymbol{y}) \sum_{i=1}^{l} \boldsymbol{x}^{(i)} \cdot \boldsymbol{y}^{(i)}$ with ΠSS requires only sending two intermediate values for the entire operation regardless of the length of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, thus reducing the communication size by a factor of l with respect to SS.

$$\begin{array}{lll}
\Pi\texttt{SS.share}(x) \rightarrow \langle\!\langle x \rangle\!\rangle & \mathsf{P}_{setup}\colon & \langle \delta_x \rangle \sim \mathcal{U}_{[\mathbb{Z}_L^2]} \\
 & & \textsc{Send}(\delta_{x_j}) \Rightarrow \mathsf{P}_j, \mathsf{P}_{in_x} \\
 & \mathsf{P}_{in_x}\colon & \Delta_x \leftarrow x + \delta_{x_0} + \delta_{x_1} \\
 & & \textsc{Send}(\Delta_x) \Rightarrow \mathsf{P}_j \\
 & & \\
\Pi\texttt{SS.reveal}(\langle\!\langle x \rangle\!\rangle) \rightarrow x & \mathsf{P}_j\colon & \textsc{Send}(\langle\!\langle x \rangle\!\rangle_j) \Rightarrow \mathsf{P}_{out} \\
 & \mathsf{P}_{out}\colon & x = \Delta_x - \delta_{x_0} - \delta_{x_1} \\
 & & \\
\Pi\texttt{SS.add}(\langle\!\langle x \rangle\!\rangle, \langle\!\langle y \rangle\!\rangle) \rightarrow \langle\!\langle x + y \rangle\!\rangle & \mathsf{P}_j\colon & \langle\!\langle x + y \rangle\!\rangle_j = \langle\!\langle x \rangle\!\rangle_j + \langle\!\langle y \rangle\!\rangle_j \\
 & & \\
\Pi\texttt{SS.mult}(\langle\!\langle x \rangle\!\rangle, \langle\!\langle y \rangle\!\rangle) \rightarrow \langle\!\langle x \cdot y \rangle\!\rangle & \mathsf{P}_{setup}\colon & \langle \delta_x \rangle, \langle \delta_y \rangle, \langle \delta_z \rangle \sim \mathcal{U}_{[\mathbb{Z}_N^{3\times 2}]}; \quad \langle \delta_{xy} \rangle \leftarrow \langle \delta_x \cdot \delta_y \rangle \\
 & & \textsc{Send}\ (\delta_{x_j}, \delta_{y_j}, \delta_{xy_j}, \delta_{z_j}) \Rightarrow \mathsf{P}_j \\
 & & \textsc{Send}\ (\delta_x) \Rightarrow \mathsf{P}_{in_x}; \quad \textsc{Send}\ (\delta_y) \Rightarrow \mathsf{P}_{in_y} \\
 & \mathsf{P}_{in_x}\colon & \Delta_x \leftarrow x + \delta_{x_0} + \delta_{x_1}; \quad \textsc{Send}\ (\Delta_x) \Rightarrow \mathsf{P}_j \\
 & \mathsf{P}_{in_y}\colon & \Delta_y \leftarrow y + \delta_{y_0} + \delta_{y_1}; \quad \textsc{Send}\ (\Delta_y) \Rightarrow \mathsf{P}_j \\
 & \mathsf{P}_j\colon & \langle x \cdot y \rangle_j \leftarrow j \cdot \Delta_x \Delta_y - \Delta_x \langle \delta_y \rangle_j - \Delta_y \langle \delta_x \rangle_j + \langle \delta_{xy} \rangle_j \\
 & & \langle \Delta_z \rangle \leftarrow \langle x \cdot y \rangle + \langle \delta_z \rangle; \qquad \textsc{Send}\ (\Delta_{zj}) \Rightarrow \mathsf{P}_{1-j} \\
 & & \langle\!\langle z \rangle\!\rangle_j \equiv (\Delta_{z0} + \Delta_{z1}, \langle \delta_z \rangle_j)
\end{array} \tag{3.3}$$

3.2.4 Functional Secret Sharing (FSS).

A 2PC Functional Secret Sharing (FSS) scheme [39, 40] for a function family $\mathcal{F}$ splits a function $f \in \mathcal{F}$ into two additive shares (f_0, f_1), such that each f_j hides f and $f_0(x)+f_1(x) = f(x)$ for every input x. Beyond trivial solutions such as secret-sharing the truth-table of

[4]Following [193], we can inject correlation from a Beaver triple in the δ-shares

[4]Note: ABY2.0 [193] refers to arithmetic secret sharing as $[\cdot]$-sharing and Π-secret sharing as $\langle\cdot\rangle$-sharing.

f, FSS schemes seek succinct descriptions of f_j (function keys $\boldsymbol{k}_0, \boldsymbol{k}_1$) with efficient online execution. Since both function shares must evaluate on the same value x, this value must be made public to both computing parties P_j. To maintain input data privacy, a random mask r is added to the secret input x, so that the opened value $\hat{x} = x + r$ completely hides x before using it as input to the FSS evaluation. In order to obtain full correctness on the function evaluation with respect to $f(x)$, the class of functions $\mathcal{F}$ is restricted to $f_r(x) = f(x + r)$, where the mask is known by the dealer and used for the key generation.

For addition and multiplication gates over a ring $\mathbb{Z}_{2^n}$, the FSS gates correspond to Beaver's protocol [22]. A much more interesting case arises in [41, 38], where non-linear operations including zero-test, integer comparison or bit decomposition are efficiently constructed using a small number of invocations of FSS primitives. Luckily, these FSS gates make a black-box use of any secure PRNG, which can be instantiated with AES blocks to yield short keys and fast implementations.

Grounded on the MPC preprocessing model, a FSS gate is composed of two algorithms:

- $\texttt{Gen}(1^\lambda,\ f) \rightarrow (\boldsymbol{k}_0,\ \boldsymbol{k}_1)$ is a PPT key generation algorithm that, given the security parameter λ and the description of a function $f : \mathbb{G}_{in} \mapsto \mathbb{G}_{out}$, outputs a pair of functional keys $(\boldsymbol{k}_0,\ \boldsymbol{k}_1)$ containing the descriptions for f_0, f_1 and the input mask shares r_0, r_1 respectively.

- $\texttt{Eval}(j,\ \boldsymbol{k}_j,\ \hat{x}) \rightarrow f(x)$ is a polynomial-time deterministic algorithm that, given the party index j, the functional key k and the masked input $\hat{x}$ outputs an additive share $f_j(\hat{x})$, such that $f_0(\hat{x}) + f_1(\hat{x}) = f(x)$.

As central building block of many FSS gates, we recall the concept of Distributed Comparison Function (DCF) (Section 3 of [38]) to be a comparison function $f^{<}_{\alpha,\beta}$ outputting β if $x > \alpha$ and zero otherwise. Built on top of two evaluations of DCF, [38] later proposes a FSS gate for Interval Containment (IC) computing $f_{p,q}(x) = \mathbb{1}_{x\in[p,q]}$ (Section 4.1 of [38]). To compute the unit step function of a n-bit signed integer, it suffices to employ their construction (detailed in Figure 3 if [38]) setting $p = 0$ and $q = 2^{n-1} - 1$, obtaining $\mathbb{1}_{p \leqslant x \leqslant q} = H_n(x)$. For convenience, we detail this FSS gate instantiation in Protocols 1 (key generation) and 2 (evaluation), keeping the DCF calls to the original protocol in [38].

For further information about FSS variants and practical gates we refer the avid reader to [38].

3.3 Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption (FHE) allows a computing party to perform computations on encrypted data without learning the inputs or the computation results. In contrast to *partially* homomorphic encryption, which supports only one type of arithmetic operation

[5]The correction terms test three standard overflow cases and one corner case. The standard case terms test if $q+r$ overflows ($\mathbb{1}_{p+r>q+r}$), if $q+r+1$ overflows ($\mathbb{1}_{q+r+1>q+1}$), and if $p+r$ does not overflow ($\mathbb{1}_{p+r>p}$, which is always 1 in our instantiation since $p = 0$ and $r < 2^n - 1$). The corner case term tests whether $p + r = 2^n - 1$ ($\mathbb{1}_{p+r=2^n-1}$, yielding zero except if $r = 2^n - 1$ in our case). Proofs of the need of these correctness terms are given in [38].

Protocol 1 $\texttt{FSS.Gen}^{IC}(\lambda,\, n,\, r) \to \boldsymbol{k}_0^{IC}, \boldsymbol{k}_1^{IC}$

Players: P_{setup} carries out the generation.
Input: λ: computational security parameter.
r: Mask for the input to the function.
Output: $\boldsymbol{k}_0, \boldsymbol{k}_1$: preprocessing keys, to send to P_0, P_1 respectively.
$\langle \delta_{\boldsymbol{x}} \rangle, \langle \delta_{\boldsymbol{y}} \rangle$: δ-shares of input vectors, to send to $\mathsf{P}_{in_{\boldsymbol{x}}}, \mathsf{P}_{in_{\boldsymbol{y}}}$ (input owners) resp.
Note: All arithmetic operations $(+,-,\cdot)$ are defined in $\mathbb{Z}_{2^n}$, thus their results are susceptible to "overflow" due to modular reduction.

```
Define the interval [p, q] for sign extraction:
1: $p \leftarrow 0$;      $q \leftarrow 2^{n-1} - 1$
Generate a DCF for γ, am arbitrary value above the two interval limits:
2: $\gamma \leftarrow (2^n - 1)$
3: $(\boldsymbol{k}_{\gamma 0}, \boldsymbol{k}_{\gamma 1}) \leftarrow \texttt{Gen}_n^{<}(1^\lambda, \gamma + r, 1, \mathcal{U}_{[\mathbb{Z}_{2^n}]})$
Generate the correction terms[5] to fix overflows:
4: $c \leftarrow 1_{p+r>q+r} + 1_{q+r+1>q+1} - 1_{p+r>p} + 1_{p+r=2^n-1}$
5: $c_0 \sim \mathcal{U}_{[\mathbb{Z}_{2^n}]}$;   $c_1 \leftarrow c - c_0$
Compose keys:
6: $\boldsymbol{k}_0^{IC} \leftarrow (\boldsymbol{k}_{\gamma 0}, c_0)$;      $\boldsymbol{k}_1^{IC} \leftarrow (\boldsymbol{k}_{\gamma 1}, c_1)$
7: return $\boldsymbol{k}_0^{IC}, \boldsymbol{k}_1^{IC}$
```

Protocol 2 $\texttt{FSS.Eval}^{IC}(j,\, \boldsymbol{k}_j,\, \hat{x}) \to o_0, o_1$

Players: P_j the selected computing party j.
Input: j: The party number, $j \in \{0, 1\}$.
$\boldsymbol{k}_j$: The key for P_j, composed of a DCF key for γ and a correction share c_j.
$\hat{x}$: Masked public input, result of reconstructing $x + r$.
Output: o_0, o_1: Additive secret shares of $1_{x \in [0, 2^{n-1}-1]}$.

```
Define the interval [p, q] for sign extraction:
1: $p \leftarrow 0$;      $q \leftarrow 2^{n-1} - 1$
Deserialize key and obtain local overflow term η:
2: $(\boldsymbol{k}_{\gamma j}, c_j) \leftarrow \boldsymbol{k}_j$
3: $\eta \leftarrow 1_{\hat{x}>p} - 1_{\hat{x}>q+1}$
Evaluate the DCF with two inputs and compute result:
4: $o_j^L \leftarrow \texttt{Eval}_n^{<}(j, \boldsymbol{k}_{\gamma j}, 1, \hat{x} - 1)$
5: $o_j^R \leftarrow \texttt{Eval}_n^{<}(j, \boldsymbol{k}_{\gamma j}, 1, \hat{x} - q)$
6: return $o_j \leftarrow j \cdot \eta - o_j^L + o_j^R + c_j$
```

(e.g. only additions or only operations), *fully* homomorphic encryption allows encrypted multiplications and additions, theoretically allowing private computation of arbitrary functions. This powerful concept was first conceived by Rivest et al. in the 1970s [210]. However, it remained unrealized until Craig Gentry presented a first feasible FHE scheme in 2009 [109]. Since then, FHE has gone from theoretical breakthrough to practical deployment. While implementations of the first FHE schemes required 30 minutes to compute a multiplication between two encrypted values, this has since dropped down to less than 20 milliseconds [143]. Even with these significant improvements, FHE multiplications are still around seven orders of magnitude slower than native CPU integer multiplication instructions. Therefore, FHE is not a drop-in replacement for existing applications and instead requires, like other secure computation solutions, that applications be specifically adapted and optimized.

3.3.1 FHE Preliminaries

A *homomorphic* encryption scheme allows computation over the ciphertexts that results in ciphertexts encrypting the result of the equivalent plaintext operation. For example, in an *additively* homomorphic scheme we get $\mathrm{Dec}\left(\mathrm{Enc}(a) \oplus \mathrm{Enc}(b)\right) = \mathrm{Dec}\left(\mathrm{Enc}(a+b)\right)$, where $+$ is the addition operation over plaintexts, and $\oplus$ is an operation over the ciphertexts. Similarly, a *multiplicatively* homomorphic encryption scheme supports multiplications over encrypted values: $\mathrm{Dec}\left(\mathrm{Enc}(a) \otimes \mathrm{Enc}(b)\right) = \mathrm{Dec}\left(\mathrm{Enc}(a \times b)\right)$ where $\times$ is the multiplication operation over the plaintexts and $\otimes$ is an operation over the ciphertexts. Additively or multiplicatively homomorphic encryption schemes have been known since the beginning of public-key cryptography. For example, textbook RSA [209] is multiplicatively homomorphic, while the Paillier encryption scheme [191] is additively homomorphic. *Fully* homomorphic encryption, in contrast, is both additively and multiplicatively homomorphic.

Because multiplication and addition can emulate `AND` and `OR` gates, respectively, over binary plaintexts, fully homomorphic encryption allows arbitrary computations to be performed, making it significantly more powerful than the previous *partially* homomorphic encryption schemes. In recent years, several breakthroughs and advancements have led to a leap in general performance and a variety of efficient schemes [42, 100, 43, 60, 65, 67] that target slightly different settings.

3.3.2 FHE Schemes

Virtually all modern FHE schemes are based on (variants of) the Learning with Errors (LWE) hardness assumption [204] and rely on a small amount of *noise* added during encryption to guarantee security. During homomorphic operations, this noise grows. This effect is negligible for additions, but very significant for multiplications. Should the noise grow too large, decryption would no longer produce correct results. Theoretically, a technique known as *bootstrapping* can be used to homomorphically reset the noise in a ciphertext. However, this is computationally very expensive for the FHE schemes we consider in this thesis, and therefore not frequently used in practice. Instead, some schemes are generally instantiated with carefully selected parameters that are large enough to allow the computation to complete without requiring bootstrapping[6].

We now introduce the Brakerski/Fan-Vercauteren (BFV) [42, 100] and Cheon-Kim-Kim-Song (CKKS) [62] schemes, foundational to our work, leaving out other schemes such as TFHE [65]. We also cover a Multi-Party variant of BFV, which will be instrumental in a decentralized decryption protocol in Chapter 6.

[6]The TFHE [65] scheme is an exception to this, as it makes extensive use of a fast bootstrapping after each multiplication.

3.3.3 The Brakerski/Fan-Vercauteren (BFV) scheme

The Brakerski/Fan-Vercauteren scheme [42, 100] is a ring-learning-with-errors (RLWE) [180] homomorphic encryption scheme. Messages are encoded in the plaintext space $R_t = \mathbb{Z}_t[X]/(X^N + 1)$ of polynomials of degree up to $N - 1$, and then encrypted into the ciphertext space $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ with $t < q$ (typically $t \ll q$), N a power of 2 and $\Delta = \lfloor q/t \rfloor$. Simply encoding each vector element into a coefficient would lead to issues during homomorphic multiplications, and instead the vectors are mapped to polynomials using the Chinese Remainder Theorem (CRT).

3.3.3.1 Description

The BFV scheme utilizes secrets sampled from two small-normed distributions: the secret key distribution $\mathcal{S}_{[R_q]}$ with coefficients sampled from a uniform distribution $s[j] \sim \mathcal{S} \triangleq \mathcal{U}(\{-1, 0, 1\})$ so that $\text{Image}(\mathcal{S}_{R_q}) = \mathbb{Z}_{\{-1,0,1\}}[X]/(X^N + 1)$. (although in some implementations they are instead sampled from $\mathcal{U}(\{0, 1\})$), and the error distribution $\mathcal{X}_{[R_q]}$ with coefficients $e[j] \sim \mathcal{X} \triangleq \mathcal{N}_{[-B,B]}(0, \sigma)$ sampled following a discrete Gaussian with standard deviation σ truncated into $[-B, B]$ where σ and B are two cryptosystem parameters.

The security of BFV is rooted in the hardness of the *decisional-RLWE problem*, informally stated as: given a uniformly random $a \leftarrow \mathcal{U}(R_q)$, a secret $s \leftarrow \mathcal{S}_{[R_q]}$, and an error term $e \leftarrow \mathcal{X}_{[R_q]}$, it is computationally hard for an adversary that does not know s and e to distinguish between the distribution of $(sa + e, a)$ and that of (b, a) where $b \leftarrow \mathcal{U}(R_q)$. A more formal definition can be found in Section 3.1 of [100].

Homomorphic addition and multiplication operate element-wise, giving rise to the SIMD parallelism. Additionally, *rotation* operations cyclically rotate the elements inside the vectors, allowing elements originally stored at different indices (also known as “slots") to interact. Since multiplications lead to the most significant noise growth, the number of subsequent multiplications (the *multiplicative depth*) should be as small as possible.

While BFV supports bootstrapping in theory, it is slow and thus the scheme is commonly instantiated with parameters large enough to handle the noise growth result of a limited multiplicative depth. Even then, a public *relinearization* should be used between multiplications to reshape the ciphertext without changing the underlying message, lowering noise growth and ciphertext size by employing a specific public key named relinearization key (rlk).

Scheme 3 outlines the subset of algorithms conforming the BFV scheme that we will be using in this thesis.

3.3.3.2 Multi-party BFV Scheme

Multiparty Homomorphic Encryption (MHE) provides a natural extension of FHE to an N-party setting. We will focus on the Distributed BFV scheme or DBFV of [186], summarizing some of the key protocols in Scheme 4. The secret key generation is now performed by local generation of shares $\langle sk \rangle_i$, then the collective public key protocol is based on the underlying global secret key sk. To preserve practical security, sk is never reconstructed, but parties can collaborate to generate a collective public key cpk corresponding to sk. Homomorphic

Scheme 3 BFV(t, n, q, w, σ, B)

```
BFV.SecKeyGen() → sk :
    SAMPLE $s \leftarrow \mathcal{S}_{[R_q]}$
    OUTPUT $sk = s$

BFV.PubKeyGen( sk ) → pk:
    LET $sk = s$ a secret key
    SAMPLE $p_1 \leftarrow \mathcal{U}(R_q)$, and $e \leftarrow \mathcal{X}_{[R_q]}$
    OUTPUT $\mathbf{pk} = (p_0, p_1) = (-sp_1 + e,\ \ p_1)$

BFV.Encrypt( pk, m ) → c_m :
    LET $\mathbf{pk} = (p_0, p_1)$ a public key
    SAMPLE $u \leftarrow \mathcal{S}_{[R_q]};\ \ e_0 \leftarrow \mathcal{X}_{[R_q]};\ \ e_1 \leftarrow \mathcal{X}_{[R_q]}$
    OUTPUT $\mathbf{c}_m = (c_{m_0}, c_{m_1}) = (\Delta m + up_0 + e_0,\ \ up_1 + e_1)$

BFV.Add( c_a, c_b ) → c_add :
    LET $\mathbf{c}_a = (c_{a_0}, c_{a_1})$, $\mathbf{c}_b = (c_{b_0}, c_{b_1})$ two ciphertexts.
    OUTPUT $\mathbf{c}_{add} = ([c_{a_0} + c_{b_0}]_q,\ \ [c_{a_1} + c_{b_1}]_q)$

BFV.Decrypt( sk, c_t ) → m_res :
    LET $sk = s$ a secret key, $\mathbf{c}_t = (c_{t_0}, c_{t_1})$ a ciphertext.
    OUTPUT $m_{res} = \left[\left\lfloor \frac{t}{q}[c_{t_0} + sc_{t_1}]_q \right\rceil\right]_t$
```

encryption and evaluation are left untouched from the original BFV scheme (Scheme 3), whereas relinearization and key switching present specific multi-party protocols (protocols 3 and 4 of [186]).

3.3.3.3 Encoding, Packing and modular operations

Inputs to BFV.Encrypt are first to be encoded into the plaintext space R_t. We consider two main encoding techniques (see Scheme 5): *base* encoding, where a single integer fills an entire plaintext, and *packed* encoding, where a vector of integers is mapped element-wise to the coefficients of the plaintext. Figure 3.1 illustrates an example of these encodings.

The *packing* technique enables Single Instruction Multiple Data (SIMD) parallelism, making it highly efficient for applications working over larger amounts of data while supporting both additive and multiplicative homomorphic operations. Due to its practicality, it

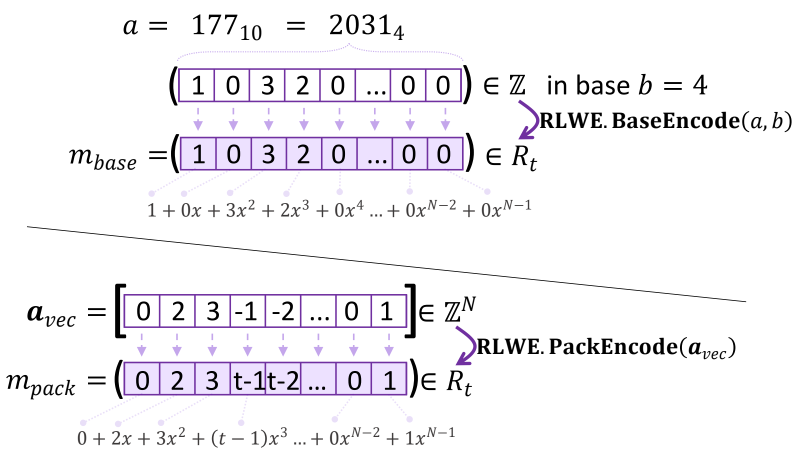
Scheme 4 DBFV(t, N, q, σ, B, K)

```
DBFV.SecKeyGen() → $\langle sk\rangle_1, \dots \langle sk\rangle_i, \dots \langle sk\rangle_K$ :
    $\mathsf{P}_i$: SAMPLE $s_i \leftarrow \mathcal{S}_{[R_q]}$
    $\mathsf{P}_i$: OUTPUT $\langle sk\rangle_i = s_i$, where $sk = \left[\sum_i^K sk_i\right]_q$

DBFV.ColPubKeyGen( $\langle sk\rangle_1, \dots \langle sk\rangle_i, \dots \langle sk\rangle_K$ ) → $\mathbf{cpk}$ :
    LET $sk_i = s_i$ private key share of $\mathsf{P}_i$.
    Any: SAMPLE $p_1 \leftarrow \mathcal{U}(R_q)$. Disclose to all $\mathsf{P}_i$.
    $\mathsf{P}_i$: SAMPLE $e_i \leftarrow \mathcal{X}_{[R_q]}$
    $\mathsf{P}_i$: COMPUTE $\langle p_0\rangle_i = -p_1 s_i + e_i$
    Any: OUTPUT $\mathbf{cpk} = (p_0, p_1) = (\sum_i^K \langle p_0\rangle_i,\ p_1)$.

DBFV.Encrypt( cpk, $m$ ) → $\mathbf{c}_m$:
    Any: OUTPUT $\mathbf{c}_m$ = BFV.Encrypt($\mathbf{cpk}, m$).

DBFV.ColDecrypt( $\mathbf{c}, \langle sk\rangle_1, \dots \langle sk\rangle_i, \dots \langle sk\rangle_K$ ) → $m_{res}$:
    LET $s_i = \langle sk\rangle_i$ private key share of $\mathsf{P}_i$, $\mathbf{c}_t = (c_{t_0}, c_{t_1})$ a ciphertext.
    $\mathsf{P}_i$: SAMPLE $e_i \leftarrow \mathcal{X}_{[R_q]}$
    $\mathsf{P}_i$: COMPUTE $\langle c_{1s}\rangle_i = s_i c_1 + e_i$
    Any: OUTPUT $m_{res} = \left[\left\lfloor \frac{t}{q}\left[c_0 + \sum_i^K \langle c_{1s}\rangle_i\right]_q \right\rceil\right]_t$
```

is implemented in most of the current lattice-based cryptographic libraries [219, 121, 196, 96] and is part of the draft HE standard [16].

Homomorphic addition is naturally performed element-wise when adding two packed polynomials. To obtain homomorphic multiplication applied element-wise, one needs to follow the instructions for RLWE-based *packing* from section 3.2 of [225]. In short, input integer vectors need to be encoded using the inverse Number Theoretic Transform (InvNTT) over R_t to turn polynomial multiplications into coefficient-wise multiplications. Additionally, *rotation* operations cyclically rotate the elements inside the vectors, allowing elements originally stored at different indices (also known as "slots") to interact.

Furthermore, typical applications of homomorphic encryption deal with operations in the non-modular domains $\mathbb{Z}$ or $\mathbb{R}$. Their coercion to arithmetic modulo t forces the underlying plaintext operations to not overflow their coefficients modulo t. Hence, the encrypted vector elements are limited to t when using *packing*, and the digits of the encrypted value in base-b



$a = 177_{10} = 2031_4$
$\in \mathbb{Z}$ in base $b = 4$
RLWE. BaseEncode(a, b)
$m_{base} =$
$\in R_t$
$1 + 0x + 3x^2 + 2x^3 + 0x^4 \ldots + 0x^{N-2} + 0x^{N-1}$
$\boldsymbol{a}_{vec} =$
$\in \mathbb{Z}^N$
RLWE. PackEncode$(\boldsymbol{a}_{vec})$
$m_{pack} =$
$\in R_t$
$0 + 2x + 3x^2 + (t-1)x^3 \ldots + 0x^{N-2} + 1x^{N-1}$

Figure 3.1: Visualization of the BaseEncode (top) and PackEncode (bottom) algorithms for arbitrary inputs

representation must fall below t when using *base* encoding. If using signed encoding, the underlying values/digits are limited to the interval $[-t/2, t/2)$. For deep arithmetic circuits, this overflow limitation causes t to take higher values, at a non-negligible performance cost. [7]

3.3.4 The Cheon-Kim-Kim-Song (CKKS) scheme

The Cheon-Kim-Kim-Song (CKKS) scheme [62] implements *approximate* homomorphic encryption, i.e., $\mathrm{Dec}(\mathrm{Enc}(m)) \approx m$. In traditional FHE schemes, interactions between the message and the noise inside a ciphertext are avoided by shifting the message to the most significant bits. Should the noise grow large enough to affect the message, this is considered an invalid ciphertext and decryption fails. In CKKS, in contrast, noise that ends up overlapping the least significant bits of a message is considered to be part of the message, leading to the approximate nature of the scheme.

CKKS is designed to be used with vectors of messages $\vec{m} \in \mathbb{R}^n$, i.e., fractional values, and encoding applies a *scaling factor*, i.e. computes $\lfloor m * \Delta \rceil$, where Δ is a large integer (e.g., 2^{30}). While this type of encoding can be used in other schemes, too, this quickly leads to issue with subsequent multiplications, as the scale will grow to Δ^2, Δ^3, etc with each multiplication. Since Δ is large, this will quickly lead to the encoded message being reduced modulo q, producing incorrect results. CKKS introduces a technique to combat this growth, introducing the ability to *rescale* a ciphertext, essentially homomorphically dividing it by Δ.

While the underlying design of CKKS is closer to the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [43] than to BFV, using CKKS is very similar from a developer perspective: homomorphic operations offer SIMD parallelism, rotations are available, and relinearization should be used after ciphertext-ciphertext multiplications.

[7]CKKS [62] solves this elegantly at the expense of introducing additional noise in the computation result.

Scheme 5 RLWE Codec(t, N)

```
RLWE.BaseEncode( $a, b$ ) $\rightarrow m$:
  LET $a \in \mathbb{Z}$ an input integer value with up to $N$ digits in base-$b$ representation.
  OUTPUT: Polynomial $m \in R_t$ with
    $m[j] = (\lfloor |a| \rfloor_{b^{(j+1)}} - \lfloor |a| \rfloor_{b^j})\ \forall j$ for unsigned encoding,
    $m[j] = (\lfloor |a| \rfloor_{b^{(j+1)}} - \lfloor |a| \rfloor_{b^j}) * sign(a) + t * sgb(a)\ \forall j$ for signed encoding.
RLWE.BaseDecode( $m, b$ ) $\rightarrow a_{res}$:
  LET $m \in \mathbb{Z}_t^N$ the coefficients of an encoded polynomial in $R_t$.
  OUTPUT Integer $a_{res} \in \mathbb{Z}$ with
    $a_{res} = \sum_{i=1}^{N} m[i] * b^{(i-1)}$ for unsigned encoding,
    $a_{res} = \sum_{i=1}^{N} m[i] * b^{(i-1)} * (-1) * sign(m[i] - t/2)$ for signed encoding.
RLWE.PackEncode( $\mathbf{a}$ ) $\rightarrow m$:
  LET $\mathbf{a} \in \mathbb{Z}^N$ an input vector with $N$ elements in:
    $[0, t)$ for unsigned encoding,
    $[-t/2, t/2)$ for signed encoding,
   where $t$ must be a prime congruent to 1 mod $2N$.
  OUTPUT Polynomial $m \in R_t$ where $m = \text{InvNTT}(\mathbf{a} \bmod t)$.
RLWE.PackDecode( $m$ ) $\rightarrow \mathbf{a}_{res}$:
  LET $m \in Z_t^N$ the coefficients of an encoded polynomial of degree $N-1$ in $R_t$.
  COMPUTE $\mathbf{a}_{dec} = \text{NTT}(m)$.
  OUTPUT
    $\mathbf{a}_{res} = \mathbf{a}_{dec}$ for unsigned encoding,
    $\mathbf{a}_{res}[i] = (\mathbf{a}_{dec}[i] - t)$ if $\mathbf{a}_{dec}[i] > t/2$ else$(\mathbf{a}_{dec}[i])\ \forall i$ for signed encoding.
```

Since the vast majority of DL techniques are designed for floating-point computations, which also exhibit variable precision and precision losses over time, this makes CKKS especially well suited for DL applications.

While the approximate nature of CKKS is essential for its practical applications, as it enables this efficient rescaling/rounding operation, it also introduces subtle security issues [168]. These are easy to mitigate in practice in some setting, e.g. when using CKKS to evaluate machine learning classifiers on encrypted input, we can release just the name or index of the top class (or top-k classes) instead of the exact probabilities computed. However, other settings might require releasing actual values and here, solutions are less obvious.

Much like in BFV, messages are encoded in the plaintext space $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ of polynomials of degree up to $n-1$, this time using a different isomorphism of $\mathcal{R}$ with $\mathbb{C}^{n/2}$ and an approximated mapping, and then encrypted into the ciphertext space $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ with n a power of 2 and q the product of a set of carefully chosen primes.

The CKKS scheme can sample secrets from the same distribution as BFV, and its security properties stem from the same hardness assumption of the *decisional RLWE problem*.

Scheme 6 CKKS$(n, q = [q_0 * q_1 * \cdots * q_{d+1}], w, \sigma, B)$

```
**CKKS.keygen( sk, w ) → (sk, pk):**
  SAMPLE $s \leftarrow \mathcal{S}_{[R_q]}$
  SAMPLE $p_1 \leftarrow \mathcal{U}(R_q)$, and $e \leftarrow \mathcal{X}_{[R_q]}$
  SET $\mathbf{pk} = (p_0, p_1) = (-sp_1 + e,\ \ p_1)$
  OUTPUT $(\mathrm{sk}, \mathbf{pk})$

CKKS.encr( pk, m ) → $\mathbf{c}_m$ :
  LET $\mathbf{pk} = (p_0, p_1)$ a public key
  SAMPLE $u \leftarrow \mathcal{S}_{[R_q]};\ e_0 \leftarrow \mathcal{X}_{[R_q]};\ e_1 \leftarrow \mathcal{X}_{[R_q]}$
  OUTPUT $\mathbf{c}_m = (c_{m_0}, c_{m_1}) = (q_0 m + up_0 + e_0,\ \ up_1 + e_1)$

CKKS.add( $\mathbf{c}_a, \mathbf{c}_b$ ) → $\mathbf{c}_{add}$ :
  LET $\mathbf{c}_a = (c_{a_0}, c_{a_1})$, $\mathbf{c}_b = (c_{b_0}, c_{b_1})$ two ciphertexts.
  OUTPUT $\mathbf{c}_{add} = ([c_{a_0} + c_{b_0}]_q,\ \ [c_{a_1} + c_{b_1}]_q)$

CKKS.decr( $sk, \mathbf{c}_t$ ) → $\mathrm{m}_{res}$ :
  LET $sk = s$ a secret key, $\mathbf{c}_t = (c_{t_0}, c_{t_1})$ a ciphertext.
  OUTPUT $m_{res} = \left\lfloor \frac{1}{q_{d+1}} [c_{t_0} + s c_{t_1}]_{q_{d+1}} \right\rceil$

CKKS.encode( a ) → $m$:
  LET $\mathbf{a} \in \mathbb{R}^n/2$ an input vector with $n/s$ elements.
  OUTPUT Polynomial $m \in R$ where $m = \mathrm{InvNTT}(\mathbf{a})$.

CKKS.decode( $m$ ) → $\mathbf{a}_{res}$:
  LET $m \in Z^N$ the coefficients of an encoded polynomial of degree $N-1$ in $R$.
  COMPUTE $\mathbf{a}_{dec} = \mathrm{NTT}(m)$.
  OUTPUT $\mathbf{a}_{dec}$
```

Similarly to BFV, CKKS is commonly instantiated with parameters large enough to handle the noise growth result of a limited number of multiplications. A reasoning on how to select these parameters in practice can be followed in Section 3.4 of [186].

Scheme 6 outlines the subset of algorithms conforming the CKKS scheme that are pertinent for this thesis.

3.4 Functional Encryption (FE)

First formalized in [34], a Functional Encryption (FE) scheme is an encryption scheme where the secret key is used to derive "functional" secret keys, allowing decryption for a certain function evaluation on the previously encrypted inputs without revealing anything else about these inputs. More concretely, an FE scheme enables evaluation of $F(k, x)$ given the encryption of x and a secret key sk_k for k, consisting of four probabilistic polynomial-time algorithms:

$pk, msk \leftarrow \texttt{FE.setup}(1^\lambda)$	generates a public and master secret key pair given security parameter λ
$sk_k \leftarrow \texttt{FE.keygen}(msk, k)$	generates functional secret key sk_k for k using master secret key msk
$c \leftarrow \texttt{FE.encr}(pk, x)$	encrypts message x with public key pk into ciphertext c
$z \leftarrow \texttt{FE.decr}(sk_k, c)$	evaluates $z = F(k, x)$ from c using sk_k

We can see FE as a generalization of public-key cryptography, since setting F to be the identity function $F(k, x) = x$ yields the original encrypted message as a result. In addition, FE also encapsulates several existing encryption schemes like Identity-based encryption (IBE) [33, 216], defining $F(k, x)$ to be equal to x when k belongs to an identity that is allowed to decrypt, and $\perp$ otherwise; and attribute-based encryption (ABE) [119], defining $F(k, x)$ to be equal to x if k contains attributes with permission to decrypt and $\perp$ otherwise.

Besides IBE and ABE derivatives (both ciphertext policy [25, 9] and key policy [15, 119]), there are only a handful of functions f for which practical and efficient FE schemes have been proposed. First studied as inner product predicate [148, 184] (whether the inner product of two vectors is zero or not), the inner dot product $bmx \cdot bmy$ between two vectors $bmx, bmy \in \mathbb{Z}_n$ is one of them [4]. The quadratic multi-variate polynomial evaluation $bmx \cdot Q \cdot bmy$ is another one [18].

Much like FHE, FE evaluation is designed to be computed in a single computing party. The key difference, however, is the custom decryption, which takes the form of a function evaluation evaluating the desired function over private inputs and yielding the result in the clear.

3.4.1 On FE Security guarantees

The security of FE is generally defined using game-based indistinguishability definitions (*IND*, [34, 4]) that prove Indistinguishability against Chosen Plaintext Attacks (IND-CPA secure), or with simulation-based definitions (*SIM*, Def. 2.4 of [153]) which also imply IND-CPA (*SIM⇒IND*, Remark 2.5 of [153]). We revisit this notion:

Definition 1 (Experiment $\text{Expt}_{FE,\lambda}^{ind-cpa-b}$). *Let $b \in \{0, 1\}$, F the functionality of $\mathcal{FE}$, a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ that can make indefinite key generation oracle queries, one challenge oracle query among them, and eventually, one finalize oracle query to end the experiment. $\mathcal{C}$ starts the experiment computing $(pk, msk) \leftarrow$ FE.setup(1^λ), gives pk to $\mathcal{A}$, sets $Y \leftarrow \emptyset$ and then responds to oracle queries made by $\mathcal{A}$ in the following manner:*

- Key generation oracle*: On input $k \in K$, $\mathcal{C}$ computes and returns $sk_k \leftarrow$ $\texttt{FE}$.keygen(msk, k), and stores $Y \leftarrow Y \cup \{k\}$.*
- Challenge oracle*: On input one messages $x_0, x_1 \in X$, computes and returns $c \leftarrow \mathcal{FE}$.encr$(msk, x_b)$.*

- Finalize oracle*: On input* $b' \in 0, 1$, $\mathcal{C}$ *first checks that* $\nexists k \in Y$ *s.t.* $F(k, x_0) \neq F(k, x_1)$. *If the check holds,* $\mathcal{C}$ *outputs* $(b = b')$, *otherwise it outputs* 0. *The experiment ends either way.*

Definition 2 (Indistinguishability-based Security)**.** *For functional encryption scheme* $\mathcal{FE} = (setup, keygen, encr, decr)$ *with functionality* $F(K, X)$, *we say that* $\mathcal{FE}$ *is secure against chosen-plaintext attacks (IND-CPA secure) if:*

$$\left|\Pr[\mathrm{Exp}_{\mathcal{FE},\lambda}^{ind-cpa-1}(\mathcal{A}) = 1] - \Pr[\mathrm{Exp}_{\mathcal{FE},\lambda}^{ind-cpa-0}(\mathcal{A}) = 1]\right| = negl(\lambda)$$

Where $f : \mathbb{N} \to [0, 1]$ *is* $negl(\lambda)$ iff $\forall a \in \mathbb{N}\ \exists \lambda_c \in \mathbb{N}$ *s.t.* $f(\lambda) \leq \lambda_c\ \forall\ \lambda > \lambda_c$

There is an important corollary to draw from this definition: There is no restriction to what the functional secret keys sk_k reveal about k. In the case of inner products $F(k, bmx) = bmx \cdot bmy_k$, this implies that sk_k can (and indeed does in standard FE schemes such as [10]) reveal bmy_k inside sk_k.

3.4.2 FE Schemes for Scalar Product

During the last decade a wide corpus of papers have tackled FE schemes for scalar product encryption. To make them more accessible to non-experts, the FENTEC project's open source library CiFEr [197] implements a curated selection of Inner Product Encryption (IPE) schemes in plain C language. Table 3.2 displays a comparison of these IPE schemes. The schemes can be characterized based on:

- *security guarantees* [34]: Selective security (s-IND-CPA, [4]) or strong security (IND-CPA / SIM-based).
- *multi-input* [5]: The scheme takes input vectors bmx_k and bmy_k from k multiple parties, realizing the multi-input variant of the inner product $z = \sum_k bmx_k \cdot bmy_k$, where the result is an aggregation of several dot products.
- *decentralized* [69]: Replaces the Trusted Party, in charge of the $FE.setup$ operation in the standard case, with a MPC protocol over several potentially untrusted parties.
- *function-Hiding* [153]: Adds a restriction so that the functional secret keys sk_k do not disclose the parametrization of the underlying function. For inner product schemes, this implies that sk_k hides the underlying vector y.

The wide spectrum of Inner Product schemes of Table 3.2 needs to be narrowed down in order to select the most appropriate FE scheme to secure biometric systems: *Selecting security guarantees.* The s-IND-CPA [4] security guarantee imposes the restriction to the attacker not to extract any information from calls to $FE.encr$ after submitting the challenge. However, in the natural setup of our system this restriction does not hold. Since a biometric system is expected to make several calls to $FE.encr$, latter calls would break the security of the previous ones. We thus set a security of IND-CPA so that all the verification attempts hold the security properties inherited from the FE scheme.

Table 3.2: Comparison of several Inner Product FE schemes based on their properties

FE scheme	Multi-input	Decentralized	Function-Hiding	Sec. guarantees
ABCP [4]	-	-	-	s-IND-CPA
ACFGU [5]	✓	-	-	s-IND-CPA
ALS [10]	-	-	-	IND-CPA
ACFGU [5]	✓	-	-	IND-CPA
CDSGPP [69]	-	✓	-	IND-CPA
ABKW [3]	✓	✓	-	IND-CPA
KLMMRW [153]	-	-	✓	SIM-based
DOT [85]	✓	-	✓	SIM-based

Selecting multi-party features. Our system does not require multi-input, as the inner product $bmx \cdot bmy$ is computed each time between a single pair of templates. We also discard the decentralized property. The enrollment phase already involves sensitive data and requires trust (otherwise the template database could be poisoned, beating the whole purpose of the system). We thus will be able to rely on the party in charge of the enrollment to act as a trusted authority.

Choosing function-hinding. Finally, we recall that the standard FE schemes do not provide privacy of the function $f(x, y)$ in the functional secret keys sk_i. For biometric matching this lack of privacy translates into sk_i yielding full information about one of the two templates involved in the matching. To address this, we require our system to have the function-hiding property [153], meaning that the scheme will effectively hide both templates: the ciphertext c and the functional secret key sk_i. With all these considerations in mind, we select the Function-Hiding Inner Product Encryption (FHIPE) scheme of [153] to be suitable for biometric systems. Contrary to standard non-function-hiding schemes, this two-input scheme uses the same key (the master secret key msk) for the encryption and for the function key generation. These are its four algorithms:

$pk, msk \leftarrow \texttt{FHIPE.setup}(1^\lambda)$	generate public parameters pp and master secret key msk given security parameter λ
$sk_y \leftarrow \texttt{FHIPE.keygen}(msk, bmy)$	generates functional secret keys sk_y for input y using master secret key msk
$c_x \leftarrow \texttt{FHIPE.encr}(msk, bmx)$	encrypts message x with master secret key msk into ciphertext c
$z \leftarrow \texttt{FHIPE.decr}(pp, sk_y, c_x)$	evaluates $z = bmx \cdot bmy$ from ciphertext c_x and functional secret key sk_y using pp

3.5 Discussion on privacy-preserving techniques

We now discuss the strengths and weaknesses of the privacy-preserving techniques we have introduced in this chapter. We also discuss the available implementations of these techniques, and hint how they can be used to build privacy-preserving biometric solutions. Table 3.3 summarizes the main points of this discussion.

On performance and types of operations. The privacy-preserving techniques can be benchmarked by comparing the amount of computation and communication each technique requires to perform a given task. We can make a clear distinction between two types of operations:

- *Linear operations*, such as addition and multiplication, applicable to all the techniques we have covered. MPC techniques suffer from a non-negligible communication overhead during the execution of multiplications, but in exchange the local computation is very efficient. FHE and FE do not involve intermediate communication, but in exchange demand more computation to perform all linear operations due to their inner mathematical structures (lattices in FHE [101, 62], bilinear pairings in FE [153]), FE being more costly than FHE.

- *Non-linear operations*, such as comparison and $\arg\max$, which are only applicable to MPC (using number-theoretical tricks as in [240], conversions between arithmetic and binary sharing as in [185], or resorting to FSS [40]) and FHE (employing polynomial approximations). The FE schemes surveyed do not support non-linear operations.

While FHE and FE are computed locally in a machine, the communication cost of sharing the inputs and outputs (e.g., sending the outputs of `FE.encrypt` or FHE.encrypt to the computing party P_j) is not negligible due to the size of their respective ciphertexts. In contrast, MPC requires communication to execute multiplications, but the communication cost of sharing the inputs and outputs is negligible.

On security guarantees and the adversary. Following the taxonomy from [73], we can classify the security guarantees of secure computation protocols into:

- Private computation (R1): parties cannot extract any information from the computation. It is the weakest security guarantee, addressing a semi-honest adversary.

- Correctness with Abort (R1)-(R4): if a maliciously corrupted party deviates, honest parties detect it and halt the computation.

- Correctness with Public Verifiability (R1)-(R4): honest parties can identify which corrupted parties (if any) are deviating from the protocol and abort.

- Correctness with Robustness (R1)-(R4): the computation is ensured to yield the correct output. Also known as Guaranteed Output Delivery.

FHE and FE provide only private computation out of the shelf, being naturally vulnerable to malicious corruptions. In contrast, MPC can provide all levels of security guarantees depending on the technique used. In this thesis we employ RSS to enhance the security guarantees of some of our privacy-preserving biometric protocols to cover not only semi-honest adversaries, but also Abort against malicious adversaries, addressing (R4)/(CH3).

We assume a static adversary throughout the thesis. We bound the computing capabilities of adversary to provide *computational hiding* for all the techniques relying on the non-invertibility of some hard problem (e.g., decisional RLWE for CKKS & BFV, bilinear pairings in the FHIPE of [153]) or the indistinguishability of the shared material from random shares (e.g., the FSS keys being indistinguishable from random values), addressing an unbounded adversary only in the case of some MPC techniques (SS, RSS, ΠSS) to obtain *perfect hiding*.

Table 3.3: Strengths and weaknesses of privacy-preserving computation techniques.

Scheme	**Type**	**Security**	**Strengths**	**Weaknesses**	**Libraries**
SS	MPC	Private computation (perfect hiding)	Simple math. For-free local addition	Requires intermediate comm. for multiplications. Non-linear functions are comm. intensive.	MP-SPDZ [149] CrypTFlow [158]
RSS	3PC	Abort against one malicious corruption (perfect hiding)	Same as SS. Tackles a malicious adversary.	Same as SS. Requires duplication w.r.t. SS. One additional round of communication for Abort.	MP-SPDZ [149] ABY3 [185] Falcon [240]
ΠSS	2PC	Private computation (perfect hiding)	Same as SS. Low intermediate communication for scalar product.	Same as SS. Circuit-dependent input sharing.	ABY2 [193]
FSS	2PC	Private computation (computational hiding, FSS keys $\overset{c}{\equiv}$ random)	Local comparison on public input	Requires same input for both parties. Heavy pre-processing material.	Splinter [241] AriaNN [213]
BFV	FHE	Private computation (computational hiding, decisional RLWE problem)	SIMD operations. Local addition and mult.	Limited to integers, limited multiplication depth. Costly Number-Theoretic comparison [138] Key management for decryption.	SEAL [219], Helib [121], OpenFHE [11], Lattigo [96]
CKKS	FHE	Private computation (computational hiding, decisional RLWE problem)	SIMD operations on Reals. Local addition and mult.	Complex lattice-based math. Extra error from approximated operations. Costly approximated comparison [164] Requires sensitive key management for decryption.	
FHIPE	FE	Private computation (computational hiding, pairing inversion problem)	Local evaluation of SP. Decryption without sensitive keys.	Complex bilinear pairings math. Comparatively slow. No support for non-linear functions.	CiFEr [197]

On strengths and weaknesses. Table 3.3 highlights the most relevant strengths and weaknesses of the techniques we have covered:

- SS, RSS, ΠSS and FSS are all MPC techniques and thus require at least two computing parties to operate. Except for FSS, all these techniques employ "simple" operations, allowing for free local addition, but requiring intermediate communication for multiplications and non-linear functions. RSS deals with a malicious adversary in 3PC in exchange of duplicated computation/communication and an additional round of communication for Abort. ΠSS presents lower intermediate communication requirements for scalar product by leveraging on circuit-dependent input sharing. FSS allows for local non-linear function evaluation on a public input by the use of heavy pre-processing material.

- BFV and CKKS are FHE schemes rooted in lattice-based cryptography providing SIMD operations, local addition and multiplication, but with limited multiplication depth, costly non-linear functions (e.g., based on Number-Theoretic tricks for BFV [138] and based on polynomial approximation for CKKS [164]), and they entail managing secret keys for decryption.

- Finally, the FHIPE scheme is a FE technique based on bilinear mathematics that permits local evaluation of a scalar product and decryption of the result without the need of key management. However, it doesn't support non-linear functions.

On the implementations. The availability of implementations is a key factor to consider when choosing a privacy-preserving technique, as applied cryptography should avoid reinventing the wheel if possible. Below we list available open-source implementations for these techniques, commenting on their functionalities:

- **MPC**: MP-SPDZ [149] remains today as the most popular open-source implementation of MPC. It supports a wide range of protocols, covering an ample set of operations, including linear and non-linear operations, with syntax close to that of Python and its own virtual environment generation to translate a protocol specification for one party into a full program for all parties. It is available for Linux and MacOS, and can be used to implement biometric protocols in a straightforward way. There are other implementations of MPC, such as ABY3 [185], Falcon [240], or Cryptflow [158], but they are not as mature as MP-SPDZ and only cover their own protocols. One small shortcoming of MP-SPDZ is that it does not support FSS, forcing us to either rely on implementations such as [241] or [213], or to roll out our own implementation.

- **FHE**: There are many implementations of FHE, but the most popular ones for BFV and CKKS schemes are HElib [121], SEAL [219] and PALISADE ([196], now migrated to OpenFHE [11]). They are all written in C++, and are widely used in the FHE literature. We can also highlight lattigo [96] for a distributed version of BFV/CKKS, and concrete [66] for a variant of TFHE (out of the scope of this thesis). However, none of them have tackled the approximation of non-linear functions in FHE.

- **FE**: With more limited coverage than FHE, the main implementation of FE is the one from [197], written in C and available for Linux. We are particularly interested on their implementations of Function-Hiding Inner Product Encryption (FHIPE) as a main building block for our privacy-preserving biometric protocols.

Almost all the privacy-preserving libraries surveyed above require a significant amount of effort to be used in practice, with implementations based on compiled languages such as C, C++, Go and Rust. This can be a major drawback to make privacy-preserving techniques available to the general public. Adopting more gentle (and more popular) programming languages such as Python would be a step forward in this direction (CH5). It is with this purpose in mind that we kick-off the first novel contribution of this thesis, the development of a Python library for FHE named PYFHEL [136] that bridges the gap between technical FHE libraries written in C++ and easy-to-use, easy-to-install tools that dominate the Python ecosystem. We dedicate Section 3.6 to this.

3.6 Pyfhel: Python for Homomorphic Encryption Libraries [136]

Abstract. Fully Homomorphic Encryption (FHE) allows private computation over encrypted data, disclosing neither the inputs, intermediate values nor results. Thanks to recent advances, FHE has become feasible for a wide range of applications, resulting in an explosion of interest in the topic and ground-breaking real-world deployments. Given the increasing presence of FHE beyond the core academic community, there is increasing demand for easier access to FHE for wider audiences. Efficient implementations of FHE schemes are mostly written in high-performance languages like `C++`, posing a high entry barrier to novice users. We need to bring FHE to the (higher-level) languages and ecosystems non-experts are already familiar with, such as Python, the de-facto standard language of data science and machine learning. We achieve this through wrapping existing FHE implementations in Python, providing one-click installation and convenience in addition to a significantly higher-level API. In this section, we present Pyfhel, introduce its design and usage and highlight how its unique support for accessing low-level features through a high-level API makes it an ideal teaching tool for lectures on FHE. In contrast to other similar works, Pyfhel goes beyond merely exposing the underlying API, adding a carefully designed abstraction layer that feels at home in Python.

3.6.1 Introduction

FHE is practical for a wide range of applications across multiple domains, and is starting to be deployed in widely used mainstream software. For example, Microsoft's Edge browser uses FHE in its privacy-preserving password monitor [147, 60, 59], which compares users' login information to known leaks without revealing users' sensitive information to the service. In the medical domain, there has been significant work on using FHE to enable large-scale genome-wide association studies (GWAS) [152, 27]. In the domain of machine learning, FHE has been applied to train, e.g., logistic regression [151, 51] models in addition to a significant body of work focusing on privacy-preserving inference for neural networks [53, 117, 83, 68]. FHE-based secure computation solutions have generated a significant amount of commercial interest and Gartner projects [91] that "by 2025, at least 20% of companies will have a budget for projects that include fully homomorphic encryption."

Given the increasing presence of FHE beyond the core academic community, we must provide easy access to FHE for a wider audience. While there exists a variety of high-quality open-source implementations of different modern FHE schemes [121, 219, 196, 65, 66], these are mostly written in `C++` or other high-performance systems languages, in a common approach for cryptographic code seeking maximal performance for heavy-weight operations. However, languages like `C++` are significantly less popular [226], especially outside of the core computer science community, than higher-level languages like Python, which has established itself as the de-facto standard language of data science and especially machine learning.

In the interest of promoting FHE, we must bring FHE implementations into the languages and ecosystems that less technical users are already familiar with. This takes the form of *wrappers* which expose interfaces for the underlying cryptographic libraries in different languages, e.g., Python. Beyond merely providing a way to access, e.g., functionalities from a `C++` library in a different language, well-written wrappers should try to provide idiomatic ways to use the code, respecting best practices and conventions of the target language. This results in code that feels familiar to developers and allows them to correctly and efficiently use the library. Additionally, these wrappers should ideally abstract away the sometimes complex installation process of these libraries. Most existing FHE implementations require, or strongly benefit from, dependencies which need to be installed through properly configured toolchains to make the library work properly. Wrappers that handle this setup and provide an automatic one-click installation greatly improve the practical accessibility of FHE.

Our Contributions. PYFHEL provides a Python wrapper for the Microsoft SEAL [219] library, extendable to other C++ libraries, that goes beyond merely exposing the underlying API by adding a carefully designed abstraction layer that feels at home in Python. PYFHEL offers *(i)* one-click install, including of the underlying libraries, *(ii)* a high-level Python-first abstraction layer that makes working with FHE significantly easier, including *(iii)* high-level APIs for low-level functionalities not generally exposed. We show how PYFHEL can not only assist developers in exploring FHE, but also how it is particularly well suited to use in FHE education.

The rest of this work is organized as follows. Section 3.6.2 presents the design principles and architecture of PYFHEL. Section 3.6.3 demonstrates its usage for common operations, and Section 3.6.4 describes two examples on how PYFHEL can be used in teaching FHE. Section 3.6.5 presents related work. Finally, Section 3.6.6 wraps up.

3.6.2 Design

3.6.2.1 Design principles

The design of PYFHEL adheres to several key principles. In terms of programming language we rely on `C++` to preserve the high efficiency of backend libraries, and Cython [23] (a superset of `C/C++` and Python) to bridge the gap with Python, fusing `C/C++` performance with Python-like expressiveness and dynamic typing. PYFHEL features a *one-click setup* that automatically installs all backends with the library, requiring no knowledge on compilation toolchains:

```
pip install Pyfhel        # Backends inside!
```

At a high level we opt for a *centralized* approach (Figure 3.2 (top)), where a single central class holds most of the functionalities and keeps track of the objects that rarely change after setup, including contexts and keys. Whereas the functional approach (Figure 3.2 (bottom)) common in FHE implementations requires the user to manually understand and keep track of the appropriate context and order of API calls, the centralized design style of Pyfhel understands and watches the global state of the FHE scheme and can raise informative errors to guide users towards the proper way to use the library, or even infer the missing pieces (e.g., generating a rotation key if not present when performing rotation).

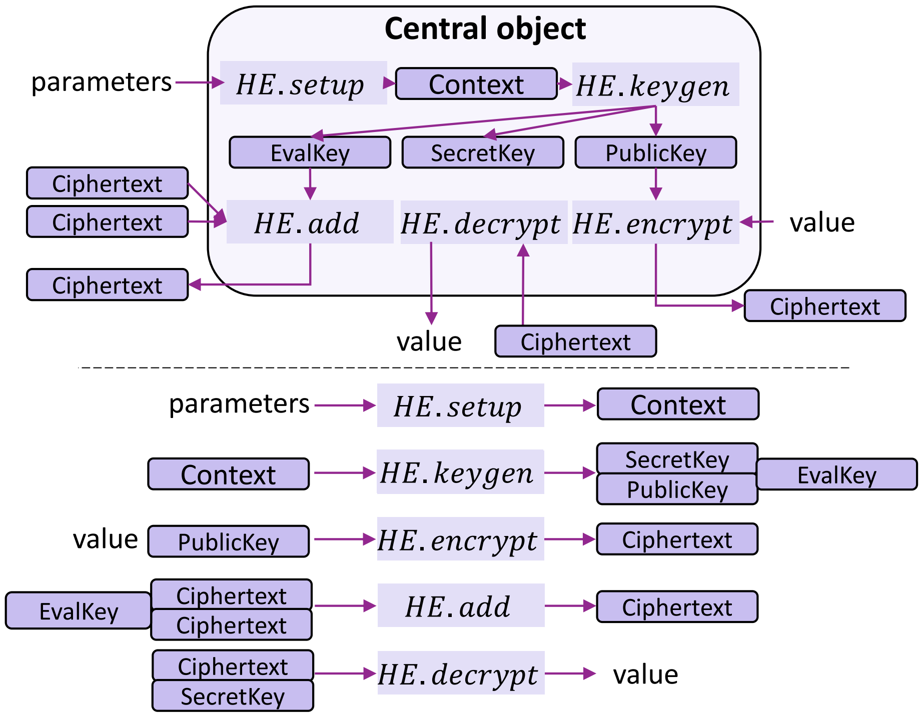


Figure 3.2: Example of centralized (top, ours) vs functional design (bottom) approaches in FHE scheme APIs

Pyfhel is also designed to cleanly expose the low-level polynomials that make up keys and plaintexts/ciphertexts. While possible in the underlying libraries, this is generally not part of the "porcelain" API intended for developers, but rather low-level "plumbing" that is documented sparingly (if at all). For example, SEAL uses a combination of custom iterator and pointer classes to deal with the underlying polynomials, manually tracking the number and sizes of the individual data elements. Pyfhel instead provides a custom high-level interface similar to that used for plain- and ciphertexts, abstracting away these implementation details.

3.6.2.2 Architecture

In order to realize these design principles, Pyfhel uses a layered architecture consisting of three key layers, as seen in Figure 3.3.

1. *Backend libraries*: The unmodified, up-to-date FHE libraries generally written in C++, automatically loaded from their official sources. These expose homomorphic operations, keys and context parameters, ciphertexts & encoded plaintexts, and serialization features. Using these correctly requires managing significant amounts of

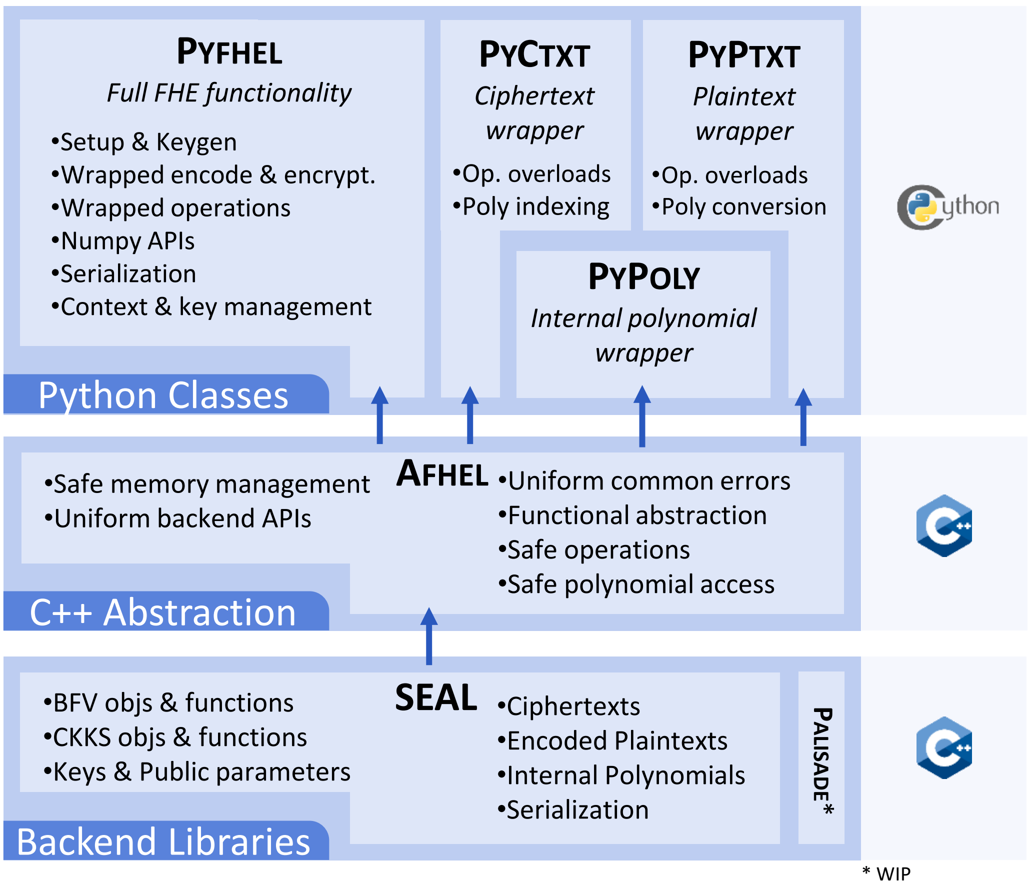


Figure 3.3: Pyfhel high level layered architecture

state information in the application code. PYFHEL currently supports SEAL [219], with work on PALISADE [196] under way.

2. *Afhel*: Our **A**bstraction **f**or **H**omomorphic **E**ncryption **L**ibs acts as a safe and uniform `C++` encapsulation of different backend APIs. In addition, it manages memory, offering memory-safe versions of low-level APIs, and tracks the state required to interact with the libraries properly. The encapsulation and abstraction offered by Afhel is key in enabling us to build clean and easy-to-use interfaces.

3. *Python Classes*: PYFHEL's python classes expose all FHE functionalities in a pythonic way, allowing users to work in a familiar setting, writing code that often looks like pseudo-code. The `Pyfhel` class centralizes a variety of functionalities, including state management, tracking keys and context required for operations on ciphertexts/plaintexts. The `PyCtxt` and `PyPtxt` classes wrap ciphertexts and plaintexts respectively, allowing users to express arithmetic expressions simply using operator overloads. These classes also offer access to the underying polynomials via simple indexing. The polynomials are wrapped by the `PyPoly` class which offers similar arithmetic operators and allows seamless conversion to and from arrays/lists of coefficients.

3.6.3 Using PYFHEL

In this subsection, we demonstrate how to use PYFHEL through a series of examples. Extended versions and additional examples documenting further features are available in the PYFHEL repository [8].

3.6.3.1 Setup and Parameters

All computations begin by creating a `Pyfhel` object, initializing a scheme with chosen parameters and generating/loading keys.

```
from Pyfhel import PyCtxt, Pyfhel, PyPtxt
HE = Pyfhel()
HE.contextGen(scheme='BFV', n=16384, p=65537)
HE.keyGen()
```

In terms of parameters, n (polynomial degree) determines the number of slots the plaintext vectors have (n in BFV and $n/2$ in CKKS). Meanwhile, p (plaintext modulus) determines the modulus of the plaintext space in BFV, which determines how large encrypted values can get before wrap-around occurs (e.g., $65537 = 2^{16}$ which is equivalent to working with 16-bit unsigned integers).

One can optionally also provide the ciphertext modulus q, which determines how much noise can accumulate before decryption fails. Larger q can tolerate more noise, and therefore more complex computations, but also lead to slower homomorphic operations and weaker security if n stays fixed. If no q is provided, `Pyfhel` uses the largest value that still achieves 128-bit security for the given polynomial degree n. Instead of providing q, advanced users can also provide a modulus chain of q_i (e.g., `qs=[30,30,30,30,30]`, which is especially useful when working with CKKS.

Although not required in normal scenarios, it is possible to set other key generation parameter, as described in the PYFHEL documentation, for expert users seeking control over lower-level aspects.

3.6.3.2 Encryption & Decryption

In order to encrypt messages, the values must first be *encoded* into plaintext objects (`PyPtxt`). Similar, after decryption, the resulting plaintext must be *decoded*. Both BFV and CKKS internally feature vector-like plaintext spaces. `Pyfhel` is able to encode a variety of different datatypes, including single values or lists/numpy arrays that are shorter than the underlying vector. In these cases, `Pyfhel` repeats the value until all slots are filled.

```
integer = 45
int_ptxt = HE.encode(integer)   # PyPtxt
int_ctxt = HE.encrypt(int_ptxt) # PyCtxt

list = [1, 2, 3, 4, 5, 6]
list_ptxt = HE.encode(list)       # PyPtxt
list_ctxt = HE.encrypt(list_ptxt) # PyCtxt

import numpy as np
```

[8]https://github.com/ibarrond/Pyfhel/tree/master/examples

```
np_array = np.array([6, 5, 4, 3, 2, 1],dtype=np.int64)
array_ptxt = HE.encode(np_array)     # PyPtxt
array_ctxt = HE.encrypt(array_ptxt) # PyCtxt

# Decrypt and Decode
ptxt_dec = HE.decrypt(int_ctxt)    # PyPtxt
integer_dec = HE.decode(ptxt_dec) # integer
```

3.6.3.3 Homomorphic Operations

The core operations of a homomorphic scheme are the addition and multiplication operation. However, there are a variety of other operations, including ciphertext maintenance operations like relinearization and rescaling. In addition to ciphertext-ciphertext operations, FHE schemes also offer ciphertext-plaintext operations that are faster and lead to noise growth. `Pyfhel` provides operator overloads for arithmetic operations ($+, -, *$ and $+=, -=, *=$ for in-place operations) which automatically select the appropriate type of operation depending on the operands. One can also use values directly in computations, with `Pyfhel` automatically encoding them into suitable plaintext objects.

```
ptxt_a = HE.encode(12)
ptxt_b = HE.encode(34)
ctxt_a = HE.encrypt(ptxt_a)
ctxt_b = HE.encrypt(ptxt_b)

# ctxt-ctxt operations
ctxt_r = ctxt_a + ctxt_b # or ctxt_a += ctxt_b (in place)
ctxt_r = ctxt_a * ctxt_b

# ctxt-ptxt operations
ctxt_r = ptxt_a + ctxt_b # or 12 + ctxt_b
ctxt_r = ctxt_a * ptxt_b # or ctxt_a * 34

# maintenance operations
HE.relinearize(ctxt_r)
HE.rescale_to_next(ctxt_r)

# rotations
ctxt_c = HE.encrypt(HE.encode([1,2,3,4]))
ctxt_rotated = ctxt_c << 1 # [2,3,4,1]
```

3.6.3.4 IO & Serialization

Pyfhel has full support for serialization, which is not only useful to store generated keys but can also be used to realize true client-server computations. In the following example, we create two independent `Pyfhel` instances, one representing the client and one representing the server. Only the client-object has access to the secret keys and can decrypt messages. For simplicity, we simulate communication using the file system, but this could easily be exchanged for a real communication channel.

```
##### CLIENT
HE = Pyfhel()
HE.contextGen(scheme='BFV', n=4096, p=65537)
HE.keyGen() # Generates public and private key
```

```
# Save context and public key only
HE.savepublicKey("mypk.pk")
HE.saveContext("mycontext.con")
# Encrypt and save inputs
ctxt_a = HE.encrypt(15) # implicit encoding
ctxt_b = HE.encrypt(25)
ctxt_a.to_file("ctxt_a.ctxt")
ctxt_b.to_file("ctxt_b.ctxt")

##### SERVER
HE_server = Pyfhel()
HE_server.restoreContext("mycontext.con")
HE_server.restorepublicKey("mypk.pk") # no secret key
# Load ciphertexts
ca = PyCtxt(pyfhel=HE_server, fileName="ctxt_a.ctxt")
cb = PyCtxt(pyfhel=HE_server, fileName="ctxt_b.ctxt")
# Compute homomorphically and send result
cr = (ca + cb) * 2
cr.to_file("cr.ctxt")

##### CLIENT
# Load and decrypt result
c_res = PyCtxt(pyfhel=HE, fileName="cr.ctxt")
print(c_res.decrypt())
```

3.6.4 Using Pyfhel in Education

In addition to allowing developers to explore FHE on their own, Pyfhel is an excellent tool for integrating FHE into teaching. Due to its one-click-install and integration into the Python ecosystem, it is much more feasible to integrate coursework based on Pyfhel into a curriculum then trying the same with the underlying `C++` libraries. By providing abstractions, syntactic sugar (e.g., operator overloads) and other conveniences, Pyfhel is considerably more concise and allows students to focus on the task at hand. Beyond providing this ease and accessibility, Pyfhel includes access to low-level features specifically tailored to enable teaching. In particular, Pyfhel allows users to easily access the underlying polynomials that actually make up plaintexts and ciphertexts. Working with the underlying polynomials is not generally necessary to employ FHE, but it can be helpful in teaching situations to be able to dissect ciphertexts and study elements individually. In the following, we present two case studies for using Pyfhel in teaching. One focuses on the challenges of managing scales in CKKS, while the other uses the polynomial API in Pyfhel to study a key recovery attack on CKKS.

3.6.4.1 Exploring common CKKS pitfalls

Implementing applications in FHE can be challenging for novice users due to a variety of common pitfalls like failing to properly manage the scaling factors throughout a CKKS-based computation. While higher-level tools and compilers like EVA [83] increasingly provide automated solutions for these challenges, asking students to tackle and predict theses issues reinforces their understanding of the scheme, since it requires transferring theoretical information about the scheme to practical application. Exploring these issues also helps illustrate the gap between FHE's theoretical power to perform "arbitrary computations" and the difficulty of developing efficient FHE-based applications in practice.

We show how to use Pyfhel to explore several pitfalls in working with CKKS, using the computation $((x + y) * (z * 5)) + 10$, where x, y and z are (secret) inputs. First, we set up context and keys:

```python
from Pyfhel import PyCtxt, Pyfhel, PyPtxt
HE = Pyfhel()
HE.contextGen(scheme='CKKS', n=16384,qs=[30,30,30,30,30])
HE.keyGen()
```

Now we can perform the ciphertext-ciphertext addition $x + y$ and the ciphertext-plaintext multiplication $z*5$ before performing the ciphertext-ciphertext multiplication between those two results.

```python
ctxt_x = HE.encrypt(3.1, scale=2 ** 30) # implicit encode
ctxt_y = HE.encrypt(4.1, scale=2 ** 30)
ctxt_z = HE.encrypt(5.9, scale=2 ** 30)

ctxtSum = ctxt_x + ctxt_y
ctxtProd = ctxt_z * 5
ctxt_t = ctxtSum * ctxtProd
```

Next, we explicitly encode the constant 10 the same as we encoded the inputs x, y and z. This will lead to an error since the scale of `ctxtProd` has increased to 2^{60} after the first multiplication, and multiplying it with `ctxtSum` which is still at scale 2^{30} (addition does not change scale) causes `ctxt_t` to have scale 2^{90}. Since, in fixed-point arithmetic, addition can only performed over numbers represented at the same scale, the addition will fail.

```python
ptxt_ten = HE.encode(10, scale=2 ** 30)
ctxt_result = ctxt_t + ptxt_ten #error: mismatched scales
```

Of course, this can be resolved by encoding 10 at the correct scale, i.e., setting `ptxt_ten =HE.encode(10, scale=2**90)`. Alternatively, we can use `ctxt_result=ctxt_t+10` and `Pyfhel` will automatically deduce the correct scale. However, if instead of a constant we have an input d that the user encrypted at the same scale as all other inputs, this solution no longer applies. Instead, we must use *rescaling* to homomorphically decrease the scale of `ctxt_t` to the initial scale. In CKKS, each rescaling reduces the scale down by one "step" (here $\Delta = 2^{30}$), so we need to perform two consecutive rescaling operations.

```python
ptxt_d = HE.encode(10, 2 ** 30)
ctxt_d = HE.encrypt(ptxt_d)
HE.rescale_to_next(ctxt_t)  # 2^90 -> 2^60
HE.rescale_to_next(ctxt_t)  # 2^60 -> 2^30
```

Surprisingly, trying to multiply `ctxt_t` and `ctxt_d` will still fail, due to a subtle issue with how rescaling is implemented in versions of CKKS that improve its efficiency [61]. In essence, rescaling also decreases the ciphertext modulus, and we need to decrease the ciphertext modulus of `ctxt_d` to match.

```python
HE.mod_switch_to_next(ctxt_d) # match first rescale
HE.mod_switch_to_next(ctxt_d) # match second rescale
```

Now, trying to compute `ctxt_t+ctxt_d` will no longer produce an error about mismatched moduli. However, it will still fail with an error about mismatched scales, even though we rescaled `ctxt_t` back down. This is, again, due to subtleties in the way rescaling works. Instead of dividing the scale by exactly the step size (2^{30}), it divides the scale by a prime number very close, but not exactly equal to, the step size. Therefore, `ctxt_t` is now actually at a scale of, e.g., $2^{29.86..}$. Since CKKS is inherently approximate to begin with, we can ignore this difference and simply accumulate it into the overall approximation error. In order to do this, we manually override the scaling factor used, which finally allows the addition to complete successfully.

```
ctxt_t.set_scale(2**30)
ctxt_result = ctxt_t + ctxt_d # final result
```

3.6.4.2 Implementing Key-Recovery for CKKS

Allowing low-level access to polynomials enables implementing a variety of advanced techniques or attacks, including the key recovery attack on CKKS by Li and Micciancio [168]. The key insight of this attack is that the *noisy* decryption reveals information about the secret key. In a non-approximate scheme, knowing the input x and the function f to be computed homomorphically allows one to perfectly simulate the computation and derive $f(x)$. However, in an approximate scheme like CKKS, the decryption will be $y \approx f(x)$ and, importantly, the differences between y and $f(x)$ depend on the secret key. Interestingly, this attack does not contradict the security guarantees proven for CKKS, as the attack is outside the IND-CPA model, and FHE schemes by definition cannot achieve IND-CCA security due to their homomorphic nature. We briefly describe the simplest form of the attack below.

In CKKS, a ciphertext ct has two components, i.e., $ct = (a, b)$ where $b = a * s + m + e$, for secret key s, random mask a and noise term e. The decryption of ct is $c := \mathrm{Dec}_s(ct) = b - a * s = m + e$. Here, $m = f(x)$, which we assume is known to the adversary. If the adversary also gains access to the decryption c, they can solve the linear equation $a*s = b-c$ by computing the multiplicative inverse a^{-1} (which exists with high probability), recovering the secret key. Note that this ignores the encoding and decoding used in CKKS. Instead of seeing the plaintext, it is more realistic to assume the attacker only has access to the decoded message. While the encoding is not actually perfectly reversible, simply re-encoding the decoded value is frequently sufficient to enable the attack.

As we can see below, implementing this attack takes only a few lines of code, using `Pyfhel`'s support for working directly with the underlying polynomials. For comparison, an equivalent `C++` implementation of this example, targeting SEAL directly, uses over a hundred lines of code and makes calls to a variety of undocumented low-level features inside SEAL.

```
# Setup: Encrypt, Decrypt, Decode
ctxt = HE.encrypt(0, scale=2**40)
ptxt_dec = HE.decrypt(ctxt)
values = HE.decodeComplex(ptxt_dec)

# Attack
ptxt_re = HE.encode(values, scale=2**40)
```

```
a = HE.poly_from_ciphertext(ctxt,1) # PyPoly
b = HE.poly_from_ciphertext(ctxt,0) # or b = ctxt[0]
m = HE.poly_from_plaintext(ptxt_re) # PyPoly
s = (m - b) * ~a # ~a = inverse of a
```

3.6.5 Related Work

There already exists a plethora of Python wrappers for FHE libraries, many of them no longer maintained and outdated. Most rely on automatic `C++` wrapping tools like `pybind11` [233, 24] or `Boost.Python` [196], which requires large parts of the wrapper logic to be written in `C++` to preserve performance. PySEAL [233] is such a no-longer-maintained `pybind11` wrapper. Many require the user to compile the underlying library themselves, using a Unix-only toolchain, like the more recent SEAL-Python [129]. TenSEAL [24], which appeared several years after the initial release of Pyfhel, shows the most promise. It is `pybind11`-based and features a one-click setup, but focuses mostly on high level Machine Learning and tensor operations. Other approaches (e.g., pyFHE [97]) implement schemes directly in Python, at the cost of significantly slower operations. Finally, FHE libraries have also experimented with Python interfaces, including PALISADE [196], and the EVA compiler [83] for SEAL. However, both still require non-python build toolchains. While TenSEAL and EVA are great for novice users, they do not offer proper access to the underlying data structures, which is required to, e.g., properly understand ciphertext maintenance in teaching settings. We argue that, just as a healthy FHE ecosystem requires different libraries implementing the same schemes, it also benefits from different ways to expose this functionality.

3.6.6 Conclusion

We have presented Pyfhel, explored its design and usage, including how it can be used as a teaching tool. By providing a python-native abstraction layer on top of existing FHE implementations, Pyfhel makes working with FHE accessible to a significantly wider audience. However, even experts can benefit from the convenience offered by Pyfhel which eliminates potential error sources and reduces time to solution.

3.7 Summary

In this chapter, we have presented the main privacy-preserving cryptographic tools we are to employ in our solutions. We started off with Multi-Party Computation, with a deep dive on Secret Sharing and some improved variants: Replicated Secret Snaring to tackle one malicious corruption in 3PC, Π-Secret Sharing for cheaper scalar products, and Functional Secret Sharing for cheap comparisons. We covered Fully Homomorphic Encryption (FHE) with the two main schemes we are to use in this thesis, CKKS and BFV, including a distributed variant of the latter (DBFV). We then surveyed Functional Encryption (FE) schemes suitable for privacy-preserving inner product operations, settling with Function-Hiding Inner Product Encryption. We discussed the strengths and weaknesses of each technique as well as available implementations, wrapping up the chapter with a presentation of our novel library named PYFHEL to address (CH5) for FHE.

Armed with these tools, we are now ready to tackle the challenges of Chapter 2:

- We devote Chapter 4 to the privacy of the feature extractor (R1), employing RSS to protect the feature extractor against one malicious corruption (CH3). We chose MPC for its relatively high performance (R6), while preserving a proper balance with privacy and accuracy (CH2).
- Chapter 5 revolves around privacy-preserving biometric verification (R1), where we examine how FHIPE (FE), CKKS(FHE) and ΠSS-FSS (MPC) can be wielded to protect the templates (R3) and their matching (R2), achieving solutions that retain a high level of biometric accuracy (R5). We will deal with the accuracy losses due to both the encoding of floating-point templates as integers and the non-linear operations with CKKS and FSS (CH1).
- Lastly, Chapter 6 studies the leakage of the biometric verification result (and how it can be leveraged to break the irreversibility (R2)), and proposes several countermeasures (CH4): limiting the number of requests to a system outputting the full matching score (e.g., the output of FHIPE), and using distributed decryption (DBFV) to minimize the leakage.

Chapter 4

Protecting the Feature Extractor

This Chapter focuses on protecting the privacy (R1) and the correctness (R4) of generic feature extractors. As we discussed in Section 2.1.4, the feature extractors employed in modern biometric systems are based on Deep Learning models with many hidden layers[1]. Inspecting our privacy-preserving toolbox from Chapter 3, we an choose among multiple MPC techniques and two FHE schemes. The use of FHE is heavily discouraged due to the naturally high multiplicative depth of these models as it would incur in a prohibitively large latency, and forced to employ bootstrapping to reset the noise level[2]. This leaves us with MPC as the only viable option for the protection of the feature extractor. However, MPC is not a panacea: they are communication-intensive techniques, and it is not trivial to find a suitable MPC scheme that is both secure and efficient (CH2). Seeking to extend the security guarantees of our system to *abort* in case of misbehavior of one party (CH3), we settle with RSS as our main building block. Henceforth, this chapter delves into the design and implementation of BANNERS, a novel maliciously secure inference protocol for Binarized Neural Networks (BNN), a binary version of the CNNs that make up biometric feature extractors.

4.1 BANNERS: Binarized Neural Networks with Replicated Secret Sharing [132]

Abstract Binarized Neural Networks (BNN) provide efficient implementations of the Convolutional Neural Networks (CNN) that conform the feature extractor. This makes them particularly suitable to perform fast and memory-light inference of neural networks running on resource-constrained devices. Motivated by the growing interest in CNN-based biometric recognition on potentially insecure devices, or as part of strong multi-factor authentication for sensitive applications, the protection of BNN inference on edge devices is rendered imperative. We propose a new method to perform secure inference of BNN relying on secure multi-party computation. While preceding papers offered security in a

[1]The open-source face recognition models from the popular Insightface repo [13] contain hundreds of layers (Conv, activations, FC, etc) each.

[2]E.g., [166] employs close to 3h for the inference of a CNN of smaller size than those found in the face recognition literature [89, 13]

semi-honest setting for BNN or malicious security for standard CNN, our work yields security with abort against one malicious adversary for BNN by leveraging on Replicated Secret Sharing (RSS) for an honest majority with three computing parties. Experimentally, we implement BANNERS on top of MP-SPDZ and compare it with prior work over binarized models trained for MNIST and CIFAR10 image classification datasets. Our results attest the efficiency of BANNERS as a privacy-preserving inference technique.

4.1.1 Introduction

Requiring orders of magnitude more data than classical Machine Learning, the recent progress in Deep Learning has attained models with near human capabilities to solve complex tasks like image classification [231], object detection [203] or natural language processing [46], also reaching unheard-of generation capabilities for text [46], audio [194] and image generation [246].

Making use of the deep learning toolbox comes at a non negligible cost: one needs to acquire very large amounts of structured data, considerable computational power and vast technical expertise to define and train the models. Subsequently, the expensively trained deep learning models can be used to perform inference on data not present during training. Naturally, risk arises when training or inference computation tasks are outsourced, following the trends of Cloud Computing (where the model is sent to the cloud) or Edge Computing (where the trained model is pushed to edge devices such as mobile phones or cars). In a standard setup, carrying out these processes on an outsourced enclave forces users to keep the model in plaintext to carry out mathematical operations, leading to potential model theft and exposing all intermediate computations as well as the input data and the inference result.

In the specific field of Biometrics there is a growing interest on using face & fingerprint recognition on potentially insecure devices [105], with applications that range from secure banking access to government services such as border control, or in general as part of strong multi-factor authentication. Addressing the protection of the underlying biometric identification algorithms on resource-constrained devices is thus rendered imperative for industry leaders in biometric solutions [137]. Since biometric feature extractors are nowadays based on modern Convolutional Neural Networks (CNN), this chapter focuses on securing the inference of a particular flavor of these networks (**R1**). Furthermore, CNN can be binarized (constrain weights and intermediate operations to 0 and 1) in order to greatly reduce the model size and memory usage, making the resulting Binarized Neural Networks (BNN) [128] suitable to execute in edge devices such as mobile phones. BANNERS serves as the first step in this direction, implementing secure BNN execution in a stronger security model (CH3).

MPC is, at the time of this writing, among the most efficient technologies providing secure outsourced computation. This chapter relies on MPC to carry out secure inference of BNNs. Going beyond the Honest-But-Curious adversary model present in many of the MPC-based secure NN inference schemes, this work uses a threat model whereby honest parties can detect a malicious adversary and abort, ensuring a correct computation if no errors are detected (R4).

Main contribution. Leaning on replicated secret sharing (RSS), Banners proposes a new method to perform secure inference of Binarized Neural Networks, guaranteeing security with abort against one malicious adversary in a 3-party setting. Throughout the chapter, this secure method is described mathematically, proven secure, implemented and compared with existing techniques. This solution is outlined as follows. Section 4.1.2 covers the preliminaries on BNN. Section 4.1.3 discusses related previous work, covering state of the art on securing CNN inference. Section 4.1.4 presents our detailed solution, covering each and every protocol we need. Section 4.1.5 describes our implementation and experiments, closing up with conclusions and future work on section 4.1.6.

4.1.2 Binarized Neural Networks

BNN [128] are a subtype of Neural Networks whose weights and activations are constrained to two values $\{-1, 1\}$ (mapped into binary values $0, 1$), taking up one bit per value while sacrificing accuracy with respect to their full precision counterparts. Thanks to this limitation, up to 64 bits can be packed together in a 64-bit register, providing high parallelization on the operations in a Single Instruction Multiple Data (SIMD) fashion. This packing technique is named *Bit Slicing* [14] [47], and it yields savings of up to 64 times in memory and space. Indeed, this makes BNN particularly suitable for edge devices and resource-constrained scenarios.

We will focus our attention on BNNs. Albeit less accurate (but recently closing the accuracy gap w.r.t. full sized models; see [223] for an in-depth comparison), they are good candidates for deep learning implementations on FPGAs and ASICs due to their bitwise efficiency. We implement all the layers of a XNOR-Net [201] architecture.

4.1.2.1 First linear layer

Linear combination of the inputs x with some weights w, there are two types of linear layers: Fully Connected (FC, also known as Dense in popular frameworks) and Convolution (Conv). FC corresponds to a matrix multiplication, whilst Conv can be turned into a matrix multiplication by applying a Toeplitz transformation on the inputs and weights. This transformation is more commonly known as `im2col` & `col2im` (more info in section 5.1 of SecureNN [239], and a nice visual explanation in slide 66 of [102]). In the end, both FC and Conv are computed as a matrix multiplication, which can be decomposed into *Vector Dot Products* (VDP). Figure 4.1 represents one VDP in the first layer of our BNN architecture, with 8-bit inputs and 1-bit weights.

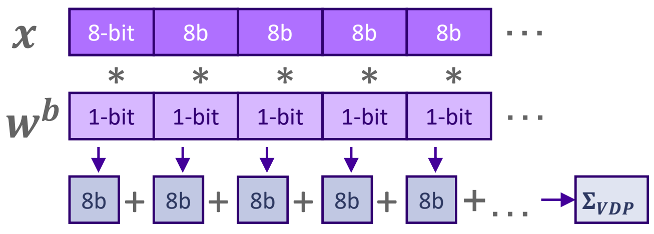


Figure 4.1: Diagram of a VDP in the first layer of a BNN

There is one peculiarity with the first linear layer of a BNN: Binarizing the input of the first layer would hurt accuracy much more than binarizing other layers in the network. Besides, the number of weights and operations in these layers tend to be relatively small. Therefore it has become standard to leave the input of this layer with higher precision (8 bits in our case).

4.1.2.2 Binary Activation and Batch Normalization

A *Binary Activation* (BA) is equivalent to the $sign(x)$ function [128], and is normally applied after a linear layer. Given that the result of the VDP in linear layers is a small integer (up to $log_2(l)$ for binary VDP and $8 \cdot log_2(l)$ for the first layer, for vectors of l elements), it is easier/faster to compute than the standard ReLU in CNNs. This functionality is implemented by extracting the most significant bit (MSB).

A *Batch Normalization* (BN) operation normalizes all the inputs by subtracting β and dividing by γ, two trainable parameters. While the original batch normalization [139] includes subtracting the mean of the input batch and dividing by its standard deviation, the binarized version can be implemented by relying solely on β and γ [201] [207]. Binary BN is most frequently located right before a BA. Together, a BN followed by a BA is equivalent to $sign(x - \beta/\gamma)$, instantiated as a comparison.

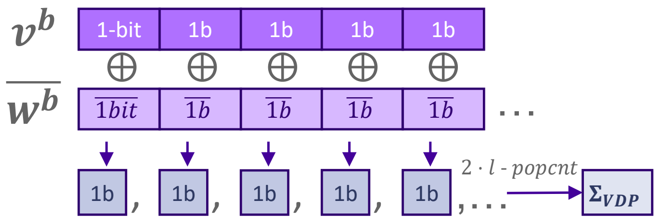


Figure 4.2: Diagram of a binary VDP

4.1.2.3 Binary linear layer

Except for the first layer, all the linear layers in a BNN have binary inputs and binary weights. Likewise, FC and Conv are turned into matrix multiplication and decomposed into a series of binary VDP. Following [201], and nicely displayed in figure 2 of XONN [207], binary VPD is equivalent to XNOR (substitute of binary multiplication) and $2 \cdot l - popcount(x)$ (analogous to cumulative addition). Thus effectively transforming $mult\&add \rightarrow XNOR\&popcount$. Figure 4.2 displays the structure of an individual binary VDP.

4.1.2.4 Maxpool layer

A maxpool layer over binary inputs is homologous to the OR operation, as shown in figure 4.3.

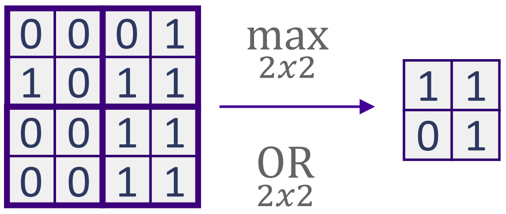


Figure 4.3: Equivalence between Binary *max* and boolean *OR* for a Maxpool layer

4.1.3 Previous Work on securing NNs

There are several publications that serve as foundations for BANNERS. The original definition of RSS is depicted in [14], with [104] adapting it to the fully malicious case. ABY3 [185] was one of the first to use RSS to secure deep neural network inference, with FALCON [240] being one of the most recent and most efficient approaches. BANNERS is inspired by certain protocols and techniques from them.

XONN [207] is the most notorious prior work addressing the subfield of secure BNN inference with MPC, relying on Garbled Circuits in a 2PC setting to secure a custom trained XNOR-Net model [201]. Consequently, we rely on the results of XONN to compare with the results of our experiments in section 5. SOTERIA [8] generalizes the Neural Architecture Search of XONN, also addressing BNN inference with GC constructions. Note that, in both cases, the security model is that of a semi-honest adversary. In contrast, our work yields security against one malicious adversary. To the best of our knowledge, this is the first work tackling maliciously secure BNN inference.

In the broader field of privacy preserving Neural Network inference, there has been a plethora of works in the recent years. A good up-to-date summary can be found in Table 1 of FALCON [240]. FHE was the foundation for the seminal CryptoNets [110] and subsequent works improved it like [53] and [125], [37] for discretized networks, [135] covering BN support for FHE and [66] perfecting programmable bootstrapping. A different line of works focused on efficient MPC implementations relying on various techniques, such as Cryptflow [158], Fantastic4 [78] and QuantizedNN [79], or hybrids using both FHE and MPC such as Gazelle [146] or Chameleon [208].

In the context of Neural Network operations, GC [244] and GMW are historically more suited for non-linear operations like comparisons, threshold-based activation functions and MaxPool, while standard (arithmetic) SS shines when used for integer addition and multiplication, which is why several previous works focused on switching between GC and SS [146] [208].

4.1.4 The BANNERS Protocol

Overall, the setting for BANNERS consists of an honest majority over 3PC, in a threat model where it provides security with abort against one malicious adversary. The next sections are tailored to these choices.

In our setup, we consider a BNN model owner, an input data owner and multiple parties/servers performing the secure computation.

We propose BANNERS which makes use of RSS to protect each of the layers described in section 2.1.4 for secure BNN inference on a 3PC (parties P_0, P_1, P_2) honest majority setting. We use binary sharing and integer/arithmetic sharing as described in [14] [104], similarly to [185] and [240].

4.1.4.1 Input data

The input data consists of a vector $x = [x_0, x_1, x_2, \ldots, x_{N-1}], x_i \in \mathbb{Z}_{2^k}$ of N integers, while the model data consists of multiple vectors of 1-bit weights $w = [w_0, w_1, w_2, \ldots, w_{k-1}], w_j \in \mathbb{Z}_2$ (being k the number of neurons at a given layer, k=N for the first layer), that can also be represented as $y = [y_0, y_1, y_2, \ldots, y_{k-1}], y_j \in \{-1, 1\}$ by the bijective mapping $y_j \leftrightarrow w_j : \{-1 \leftrightarrow 0, +1 \leftrightarrow 1\}$. We define $v = [v_0, v_1, v_2, \ldots, v_{k-1}], v_j \in \mathbb{Z}_2$ as the vector of bits used as input to an arbitrary hidden layer. The model data also includes the BN parameters γ and β, as well as the entire architecture of the model. Note that BANNERS requires the architecture (number of layers, type and configuration of each layer) to be publicly shared with all the computing parties. We do not protect against model inversion [103] or model retrieval attacks [235], as it is orthogonal to our purposes.

The protocol makes use of a secure transfer of shares from the data holders to the three computing parties/servers relying on standard secure communication protocols. Input x and all the model parameters are shared with the parties using RSS.

On the size/format of x_j and y_j Typically, the input of a CNN is an image whose values have been normalized (between 0 and 1), thus requiring float point arithmetic with sufficient decimals to maintain the accuracy of the first layer. However, the original rectangular image is made of RGB pixels taking integer values between 0 and 255 (in a 8-bit color map). Knowing this, we remove the normalization from the data preprocessing, relying on the first Batch Normalization layer to accomplish such task. The input values are set to 8-bits, and the shares of the inputs can also be set to 8 bits, minimizing the size of the communication while preserving security: $x_j \in \mathbb{Z}_{2^8}$. By additionally changing the input domain from $[0, 255]$ to $[-128, 127]$ we would be centering it on 0 while keeping the input distribution intact. We can interpret this as a scale shifting, which is translated implementation-wise into changing from unsigned integers to signed integers without modifying the values, all while using a fixed-point representation of signed (2s complement) integers in 8-bits. This proves useful when operating with the first layer weights. The first layer weights y_j take the mathematical values $-1, +1$ in the operation. While in the Binary layers we would map the y_j weights into bit values $w_j \in 0, 1$ as a result of the *mult&add* $\rightarrow$ *XNOR&popcount* transformation (see 4.1.2.3), in the first layer we are interested on keeping their mathematical representation to operate normally. We format them as 8-bit signed values, compressing them during communication into single bits $y_j \rightarrow w_j : -1 \rightarrow 0, +1 \rightarrow 1$ to reduce 8x the amount of communication (and reconstructing them upon reception $w_j \rightarrow y_j$). y_j is shared among parties using binary RSS on the bits w_j. Thanks to the bijective mapping, we preserve the same security properties present in binary RSS.

4.1.4.2 First layer VDP

To be consistent with previous work, we reuse notation from XONN [207]. XONN's linear operation in the first layer is defined as:

$$\rho_{XONN} = f(bmx, bmw) = \sum_{j=1}^{N} x_j * (-1)^{\bar{w}_j} = \sum_{j=1}^{N} x_j * y_j \tag{4.1}$$

This operation is carried out in BANNERS with local arithmetic multiplication further reconstruction of the RSS shares, ending with the local cumulative addition.

$$\begin{aligned} &\sum_{j=1}^{N} \langle\!\langle x_j \rangle\!\rangle * (-1)^{[\![w_j]\!]} \xrightarrow[\text{mult.}]{\text{local}} \sum_{j=1}^{N} \langle z_j \rangle \xrightarrow[\text{comm.}]{\text{1 round}} \sum_{j=1}^{N} \langle\!\langle z_j \rangle\!\rangle \\ &\xrightarrow[\text{cumm. add}]{\text{local}} \Sigma_{VDP}(\rho_{XONN}) \end{aligned} \tag{4.2}$$

For each individual multiplication $z_j = x_j * y_j$

$$\begin{aligned} z = x * y + 0 &= (\langle x \rangle_0 + \langle x \rangle_1 + \langle x \rangle_2) + (\langle y \rangle_0 + \langle y \rangle_1 + \langle y \rangle_2) \\ &= [(\langle x \rangle_0 + \langle x \rangle_1) * \langle y \rangle_0 + \langle x \rangle_0 * \langle y \rangle_1] + \\ &\quad [(\langle x \rangle_1 + \langle x \rangle_2) * \langle y \rangle_1 + \langle x \rangle_1 * \langle y \rangle_2] + \\ &\quad [(\langle x \rangle_2 + \langle x \rangle_0) * \langle y \rangle_2 + \langle x \rangle_2 * \langle y \rangle_0] = \\ &\quad \langle r \rangle_2 * \langle y \rangle_0 + \langle x \rangle_0 * \langle y \rangle_1 \Rightarrow \text{Locally computed in } P_0 \text{ as } \langle z \rangle_0 \\ &+ \langle r \rangle_0 * \langle y \rangle_1 + \langle x \rangle_1 * \langle y \rangle_2 \Rightarrow \text{Locally computed in } P_1 \text{ as } \langle z \rangle_1 \\ &+ \langle r \rangle_1 * \langle y \rangle_2 + \langle x \rangle_2 * \langle y \rangle_0 \Rightarrow \text{Locally computed in } P_2 \text{ as } \langle z \rangle_2 \end{aligned} \tag{4.3}$$

For the cumulative addition In order to avoid overflow in the cumulative addition we need $log_2 N$ extra bits. We cast z_j from 8-bit to either 16-bit or 32-bit (depending on the size of the VDP) and perform local addition including common randomness to hide the result from other parties:

$$\begin{aligned} \rho = \sum_{j=1}^{N} \langle\!\langle z_j \rangle\!\rangle = &\sum_{j=1}^{N} \langle\!\langle z_j \rangle\!\rangle_0 + \sum_{j=1}^{N} \langle\!\langle z_j \rangle\!\rangle_1 + \sum_{j=1}^{N} \langle\!\langle z_j \rangle\!\rangle_2 + \alpha_0 + \alpha_1 + \alpha_2 = \\ &\sum_{j=1}^{N} \langle\!\langle z_j \rangle\!\rangle_0 + \alpha_0 \Rightarrow \text{Locally computed in } P_0 \text{ as } \langle\!\langle \rho \rangle\!\rangle_0 \\ &\sum_{j=1}^{N} \langle\!\langle z_j \rangle\!\rangle_1 + \alpha_1 \Rightarrow \text{Locally computed in } P_1 \text{ as } \langle\!\langle \rho \rangle\!\rangle_1 \\ &\sum_{j=1}^{N} \langle\!\langle z_j \rangle\!\rangle_2 + \alpha_2 \Rightarrow \text{Locally computed in } P_2 \text{ as } \langle\!\langle \rho \rangle\!\rangle_2 \end{aligned} \tag{4.4}$$

As a result we obtain 2-out-of-3 shares of ρ. We describe the entire computation in algorithm 7.

Algorithm 7 Integer-binary VPD:

```
Players: $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2$ hold integer shares $\langle\langle x_j \rangle\rangle$ and $\langle\langle y_j \rangle\rangle$ in $\mathbb{Z}_{2^l}$
Correlated randomness: $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2$ hold shares of zeroed value $\langle\langle \alpha_j \rangle\rangle$.
Output: All parties get integer shares of $\langle\langle \Sigma_{VDP} \rangle\rangle$.
Note: All shares are over $\mathbb{Z}_{2^l}$, with $l$ large enough to avoid overflow (upper bound $\log_2(N)+$
    8, based on layer size).
1: $\langle z_j \rangle = \langle\langle x_j \rangle\rangle * \langle\langle y_j \rangle\rangle$
2: $\mathsf{P}_i$ sends: $\langle z_j \rangle_i \rightarrow \mathsf{P}_{i-1}$, and $\langle z_j \rangle_{i+1} \rightarrow \mathsf{P}_{i+1}$ to verify result (Figure 7 of [185])
3: if $\langle z_j \rangle_{i+1} \neq \langle z_j \rangle_{i-1}$ then
4:     abort
5: else
6:     Reconstruct $\langle\langle z_j \rangle\rangle$
7: $\langle\langle \Sigma_{VDP} \rangle\rangle_i = \sum_{j=1}^{N} \langle\langle z_j \rangle\rangle_i + \langle\langle \alpha_j \rangle\rangle$
8: return Shares of $\langle\langle \Sigma_{VDP} \rangle\rangle \in \mathbb{Z}_{2^l}$
```

4.1.4.3 BN + BA as secure comparison

Based on [201], we make use of the transformation of BN + BA into $sign(x - \rho - \beta/\gamma)$, and while the subtraction $\rho - \beta/\gamma$ can be performed locally using shares of $\langle\langle \beta/\gamma \rangle\rangle$ gotten as part of the input data, we still need a secure way to perform $q = sign(n)$. Following the $mult\&add \rightarrow XNOR\&popcount$ transformation (and its corresponding mapping $y_j \rightarrow w_j$), the $sign(n)$ function turns into $H(n)$, the Heaviside[3] function a.k.a. step function:

$$q = Heaviside(n) = H(n) = \begin{cases} 0 & \text{if } n < 0 \\ 1 & \text{if } n \geq 0 \end{cases} \tag{4.5}$$

As seen in previous work [239], this is equivalent to extract and negate the MSB in our fixed-point arithmetic representation. Indeed this is a step required to compute ReLU in FALCON [240] and SecureNN [239], which makes our activation function cheaper than standard ReLU.

Together, they turn into a comparison between input x and β/γ, implemented by extracting the MSB (the sign) of $x - \beta/\gamma$. We rely on FALCON's PrivateCompare (Algorithm 1 in [240]), simplifying it further by setting $r = 0$, described in algorithm 8. Since this algorithm requires shares of bits of x, we reuse the same constructions used in FALCON to switch from arithmetic shares $\langle\langle x \rangle\rangle_i \in \mathbb{Z}_{2^l}$ generated by linear layers to shares of bits of x in Z_p, with $p = 37$.

[3]The Heaviside function is equivalent to the derivative of ReLU $dReLU = \frac{\partial max(0,x)}{\partial x}$. The only difference is that H(t) is defined for $\mathbb{Z}$ only, while dReLU is defined for $\mathbb{R}$

Algorithm 8 Binary BN + BA:

Players: $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2$ hold binary shares of $[\![x]\!]$ in $\mathbb{Z}_2$.
Correlated randomness: $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2$ hold shares of a random bit in two rings $[\![\beta]\!]2$ and $[\![\beta]\!]p$ and shares of a random, secret integer $m \in \mathbb{Z}_p^*$.
Output: All parties get shares of the bit $(x \geq 0) \in \mathbb{Z}_2$.
Note: Arithmetic shares are over $\mathbb{Z}_p$ after conversion.

```
1: $\langle\!\langle z \rangle\!\rangle = \langle\!\langle x \rangle\!\rangle - \langle\!\langle \beta/\gamma \rangle\!\rangle$
2: **arith2bitdecomp:** (from [240]) $\langle\!\langle z \rangle\!\rangle \rightarrow$ shares of bits of $z$, $[\![z_i]\!], i \in 1, \ldots, l$
3: **for** $i = \{\ell - 1, \ell - 2, \ldots, 0\}$ **do**
4:     Compute shares of $c[i] = (-1)^{\beta} z[i] + 1 + \sum_{k=i+1}^{\ell} z[k]$
5: Compute and reveal $d := [\![m]\!]p \cdot \prod_{i=0}^{\ell-1} c[i] \pmod{p}$
6: Let $\beta' = 1$ if $(d \neq 0)$ and 0 otherwise.
7: **return** Shares of $[\![\beta' \oplus \beta]\!] \in \mathbb{Z}_2$
```

Note that, contrary to FALCON, we can directly benefit from the binary sharing returned by the private compare algorithm, since it will serve as input to subsequent binarized layers without requiring a reconversion to $\mathbb{Z}_{2^l}$.

4.1.4.4 Binary VDP

Vectorized XNOR ($bmt = bmv \oplus bm\bar{w}$) in a RSS setting is computed locally based on local XOR ([14] section 2.1) and local negation of 1 out of the 3 shares. PopCount is translated into cumulative addition by converting binary shares into arithmetic shares using the protocol in section 5.4.2 of ABY3 [185](simplified by setting $a = 1$):

$$\rho_{XONN} = f(v, w) = \sum_{j=1}^{N} v_j * (-1)^{w_j} \sum_{j=1}^{N} v_j \oplus (w_j) = \sum_{j=1}^{N} t_j \tag{4.6}$$

With the binary input vector v, and the weights vector w, we implement their VDP using XONN's *mult&add* $\rightarrow$ *XNOR&popcount* transformation:

$$\begin{aligned} &\sum_{j=1}^{N} [\![v_j]\!] \oplus [\![\bar{w}_j]\!] \xrightarrow[\text{XOR, NOT}]{\text{local}} \sum_{j=1}^{N} [\![t_j]\!] \\ &\xrightarrow[\text{comm.}]{\text{2 rounds}} \sum_{j=1}^{N} \langle\!\langle t_j \rangle\!\rangle \xrightarrow[\text{cumm. add}]{\text{local}} \Sigma_{VDP}(\rho_{XONN}) \end{aligned} \tag{4.7}$$

4.1.4.5 XNOR

Starting with 2-out-of-3 shares of a vector of bits $[\![v]\!]$, and similar shares of binary weights $[\![w]\!]$, we use local evaluation of XOR from [14] to implement XOR, where $r = [r]_0 \oplus [r]_1 \oplus [r]_2$ is the correlated randomness of v and $s = [s]_0 \oplus [s]_1 \oplus [s]_2$ is the correlated randomness of w; and $r = s = 0$. Note that, using the binary sharing proposed above, party P_i holds w_i, w_{i+1}, and thus holds $s_{i-1} = w_i \oplus w_{i+1}$; respectively for v_i, v_{i+1} and r_{i-1}

$$\begin{aligned}
&[\![t]\!] = [\![v \oplus w, r \oplus s]\!] = \\
&\quad [(\,[v]_0 \oplus [v]_1 \oplus [v]_2) \oplus ([w]_0 \oplus [w]_1 \oplus ([w]_2)), \\
&\quad (\,[r]_0 \oplus [r]_1 \oplus [r]_2) \oplus ([s]_0 \oplus [s]_1 \oplus ([s]_2)] \rightarrow \\
&\qquad [w]_0 \oplus [v]_0, [r]_2 \oplus [s]_2 \Rightarrow \text{Locally computed in } P_0 \text{ as } [\![t]\!]_0 \\
&\qquad [w]_1 \oplus [v]_1, [r]_0 \oplus [s]_0 \Rightarrow \text{Locally computed in } P_1 \text{ as } [\![t]\!]_1 \\
&\qquad [w]_2 \oplus [v]_2, [r]_1 \oplus [s]_1 \Rightarrow \text{Locally computed in } P_2 \text{ as } [\![t]\!]_2
\end{aligned} \tag{4.8}$$

4.1.4.6 Popcount

The equivalent of cumulative addition for integers, *popcount* (or hamming weight) adds all the bits set to 1. To perform this cumulative addition, standard Garbled Circuits require an entire tree of ripple carry adders (RCA) [242], as it is the case in XONN [207]. This renders the computation quite expensive, seeing how each RCA requires at least one AND operation (1 round of communication each, log_2N rounds in total). Instead, based on section 5.4.2 of ABY3 [185] we convert the binary shares into integer shares at a cost of 2 multiplications and then perform local cumulative addition over the resulting integer shares, just like in the first layer.

The conversion happens as follows:

$$\sum_{j=1}^{N} [\![t_j]\!] \xrightarrow[\text{comm.}]{\text{2 rounds}} \sum_{j=1}^{N} \langle\!\langle t_j \rangle\!\rangle \xrightarrow[\text{cumm. add}]{\text{local}} \Sigma_{VDP}(\rho_{XONN}) \tag{4.9}$$

The entire binary linear layer would look like this:

$$\begin{aligned}
\langle\!\langle b \rangle\!\rangle =& 2 * N - \sum_{j=1}^{N} [\![v_j]\!] \oplus [\![\bar{w_j}]\!] \xrightarrow[\text{XOR, NOT}]{\text{local}} 2 * N - \sum_{j=1}^{N} [\![t_j]\!] \xrightarrow[\text{comm.}]{\text{2 rounds}} \\
& 2 * N - \sum_{j=1}^{N} \langle\!\langle t_j \rangle\!\rangle \xrightarrow[\text{cumm. add}]{\text{local}} 2 * N - \Sigma_{VDP}
\end{aligned} \tag{4.10}$$

The actual output of the binary VDP is $2*\Sigma_{VDP}-N$, as shown in figure 2 of XONN [207]. The complete Binary VDP is detailed in algorithm 9.

Algorithm 9 Binary VDP:

Players: $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2$ hold binary shares of $[\![x_j]\!]$ in a given window spanning $(1 \dots j \dots N)$.

Correlated randomness: $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2$ hold integer shares of zeroed values $\langle\!\langle a_j \rangle\!\rangle, \langle\!\langle b_j \rangle\!\rangle, \langle\!\langle c_j \rangle\!\rangle, \langle\!\langle \alpha_j \rangle\!\rangle$.

Output: All parties get integer shares of Res_{VDP}.

Note: Shares over $\mathbb{Z}_{2^l}$ are defined with l large enough to avoid overflow (upper bound $\log_2(N)$, based on binary layer size). Arithmetic multiplications in steps 6 and 7 also include the **abort** mechanism from algorithm 7.

```
1: $[\![t_j]\!] = [\![v_j]\!] \oplus [\![\bar{w}_j]\!]$
2: **bin2arith** $[\![t_j]\!] \to \langle\!\langle t_j \rangle\!\rangle$
3:     $\mathsf{P}_0$: $\langle\!\langle t_j \rangle\!\rangle_{b0} = [\![t_j]\!]_0 + \langle\!\langle a_j \rangle\!\rangle$
4:     $\mathsf{P}_1$: $\langle\!\langle t_j \rangle\!\rangle_{b1} = [\![t_j]\!]_1 + \langle\!\langle b_j \rangle\!\rangle$
5:     $\mathsf{P}_2$: $\langle\!\langle t_j \rangle\!\rangle_{b2} = [\![t_j]\!]_2 + \langle\!\langle c_j \rangle\!\rangle$
6:     $\langle\!\langle d_j \rangle\!\rangle = \langle\!\langle t_j \rangle\!\rangle_{b0} + \langle\!\langle t_j \rangle\!\rangle_{b1} - 2 * \langle\!\langle t_j \rangle\!\rangle_{b0} * \langle\!\langle t_j \rangle\!\rangle_{b1}$
7:     $\langle\!\langle t_j \rangle\!\rangle = \langle\!\langle t_j \rangle\!\rangle_{b2} + \langle\!\langle d_j \rangle\!\rangle - 2 * \langle\!\langle t_j \rangle\!\rangle_{b2} * \langle\!\langle d_j \rangle\!\rangle$
8: $\langle\!\langle \Sigma_{VDP} \rangle\!\rangle = \sum_{j=1}^{N} \langle\!\langle t_j \rangle\!\rangle + \langle\!\langle \alpha_j \rangle\!\rangle$
9: $\langle\!\langle Res_{VDP} \rangle\!\rangle = 2 * N - \langle\!\langle \Sigma_{VDP} \rangle\!\rangle$
10: **return** Shares of $\langle\!\langle Res_{VDP} \rangle\!\rangle \in \mathbb{Z}_{2^l}$
```

4.1.4.7 Max pooling

Max pooling requires computing the OR function over the values in the sliding window. However, [14] only defines NOT, XOR and AND as operations in the binary sharing domain. In order to compute OR, we reformulate OR with the available gates using NAND logic and decomposing: $OR(a, b) = NOT(AND(NOT(a), NOT(b)))$. We can now formulate the Max operation that composes a Maxpool layer:

$$\begin{aligned} m =& max_{\text{window } q}(x) = x_{q_1} \; OR \; x_{q_2} \; OR \; \dots = \\ & not(not(x_{q_1}) \; AND \; not(x_{q_2}) AND \; \dots) \equiv \overline{\overline{x_{q_1}} \; \& \; \overline{x_{q_2}} \; \& \; \dots} \end{aligned} \tag{4.11}$$

As such, the Binary Maxpool layer requires as many multiplications as the number of elements in the sliding window, with 4 being a typical value. This implies one communication round per multiplication. The full layer is described in algorithm 10.

Algorithm 10 MaxPool:

Players: $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2$ hold binary shares $[\![x_j]\!]$ over a window of size $1 \dots j \dots N$

Correlated randomness: $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2$ hold binary shares of zeroed bits $[\![a_j]\!]$.

Output: All parties get binary shares of $[\![m]\!]_{maxpool}$.

Note: $\&(AND)$ operation is performed following [14], with **abort** conditions similar to those in algorithm 7, but applied in $\mathbb{Z}_2$. $\bar{b}(NOT)$ is performed locally negating the binary shares in P_0.

```
1: $[\![m]\!] = \overline{[\![x_0]\!]}$
2: **for** $i = \{2, \dots j, \dots, N\}$ **do**
3:     $[\![m]\!] = [\![m]\!] \; \& \; \overline{[\![x_j]\!]}$
4: $[\![m]\!] = \overline{[\![m]\!]}$
5: **return** Shares of $[\![m]\!] \in \mathbb{Z}_2$
```

4.1.5 Experiments with Banners

4.1.5.1 Implementation

We implemented Banners on top of MP-SPDZ [149], with our own data management functions (`im2col` [102], `col2im`, `padding`, `flatten`) while relying on existing functionalities in MP-SPDZ to handle the MPC session and its low level operations. We report the total communication and the total online processing time, purposely leaving out offline processing. We used Larq [107], a high level BNN framework extending Tensorflow [1], to define and train our own BNN models for image classification over the MNIST and CIFAR10 datasets. We relied on recommendations from [26] to define BNN architectures, and used the Bop optimizer [124] with notions from [12] for training. To compare with XONN, we applied early stopping once the accuracy reported in [207] is reached, with a maximum deviation of 0.2%. Contrary to secure non-binarized NN inference (whose floating point operations need to be translated into fixed-point [239]), the secure BNN inference performs exactly the same operations as standard BNN, preserving the model accuracy.

4.1.5.2 Comparison with XONN

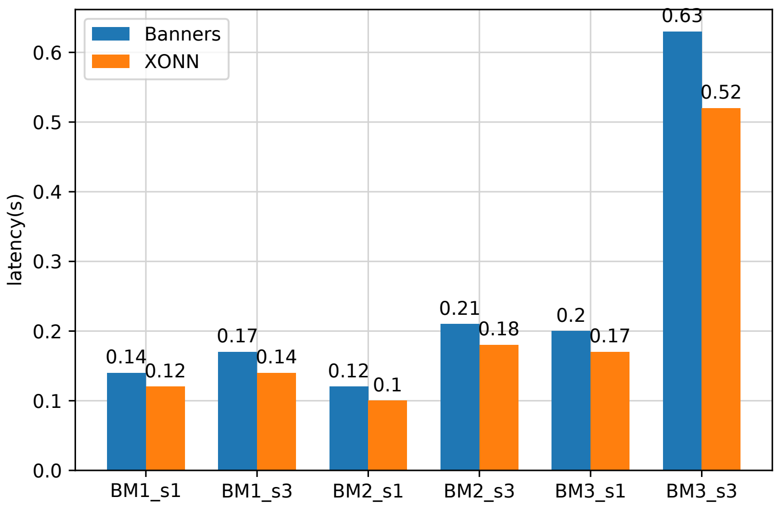


Figure 4.4: Comparison in latency for MNIST BNN models

As the most significant prior work on secure BNN inference, we chose to compare Banners with XONN [207] [4]. In order to do so, we trained the BNNs shown in table 4.1, whose architectures are directly taken from XONN([207], Appendix A.2). Since XONN defines a scaling parameter **s** that increases the number of feature maps in a given BNN, we trained models for s in $(1, 1.5, 2, 3, 4)$ to compare ourselves with tables 11 and 12 from [207].

All our experiments were ran in a simplified LAN setting (different TCP ports) on a single machine Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz with 12 cores, using 4 cores per

[4]Comparison with other non-binarized works can be inferred from tables 4 and 6 of XONN [207], as well as table 2 of FALCON [240].

Table 4.1: BNN architectures trained in Banners for comparison with XONN

Arch.	Previous Papers	#layers	Description	Dataset
BM1	XONN [207], MiniONN [172]	3	3FC	MNIST
BM2	XONN [207], CryptoNets [110], MiniONN [172]	3	1 CONV, 2 FC	MNIST
BM3	XONN [207], MiniONN [172]	6	2CONV,2MP,2FC	MNIST
BC1	XONN [207], Chameleon [208], Gazelle [146]	10	7CONV,2MP,1FC	CIFAR10
BC2	XONN [207], Fitnet [211]	13	9CONV,3MP,1FC	CIFAR10
BC3	XONN [207], Fitnet [211]	13	9CONV,3MP,1FC	CIFAR10
BC4	XONN [207], Fitnet [211]	15	11CONV,3MP,1FC	CIFAR10
BC5	XONN [207], Fitnet [211]	21	17CONV,3MP,1FC	CIFAR10
BC6	XONN [207], VGG16 [224]	19	13CONV,5MP,1FC	CIFAR10

party with an enforced communication delay of 20ms on each link. Standard point-to-point secure channels are set in place using MP-SPDZ codebase.

Observing the results from figures 4.4-4.5 (detailed results in tables 4.1.5.2 for MNIST models and 4.1.5.2 for CIFAR10 models), we discover that, while the latency is increased in average 18% for the MNIST models and 27% for the CIFAR models, the communication is 5% (CIFAR10) to 15% (MNIST) lower. We can safely conclude that Banners trades some speed in exchange for slightly lower communication and a more robust security model: while XONN offers security against a semi-honest adversary, Banners can detect misbehavior and stop the computation. Furthermore, if the protocol outputs a value, then parties are inherently sure of the correctness of the output.

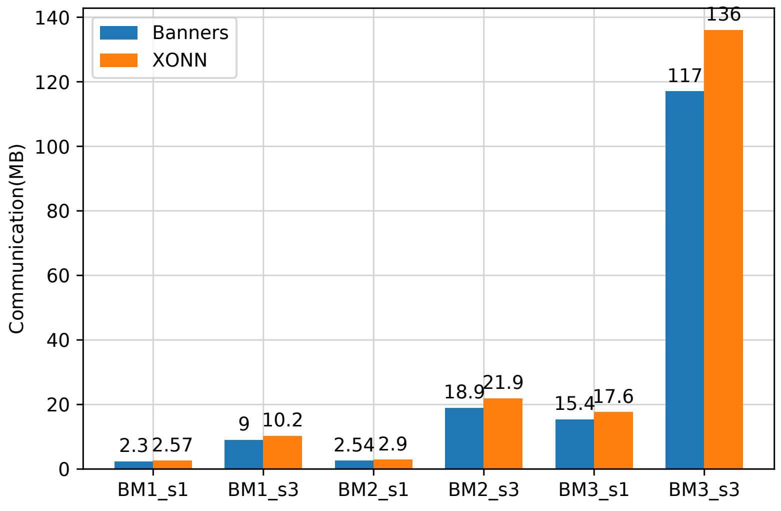


Figure 4.5: Comparison in communication for MNIST BNNs

We can bring the analysis further by comparing all the BNN models in terms of the number of Multiply-Accumulate (MAC) operations. A MAC accounts for an individual VDP operation (element-wise multiplication with cumulative addition), and given that BN and BA are applied element-wise to the output of a VDP, the number of MACs in a model is representative of its complexity[5]. The latency is higher for all BNN models in our comparison from figures 4.6 and A.3. Furthermore, while the communication increases linearly with the model complexity, there seems to be a certain inherent setup latency: note the almost horizontal slope in figure 4.7 for the smallest models, affecting both XONN and Banners models. This setup is rendered negligible when the BNN architectures increase in size, such as with CIFAR10 models in figure A.3.

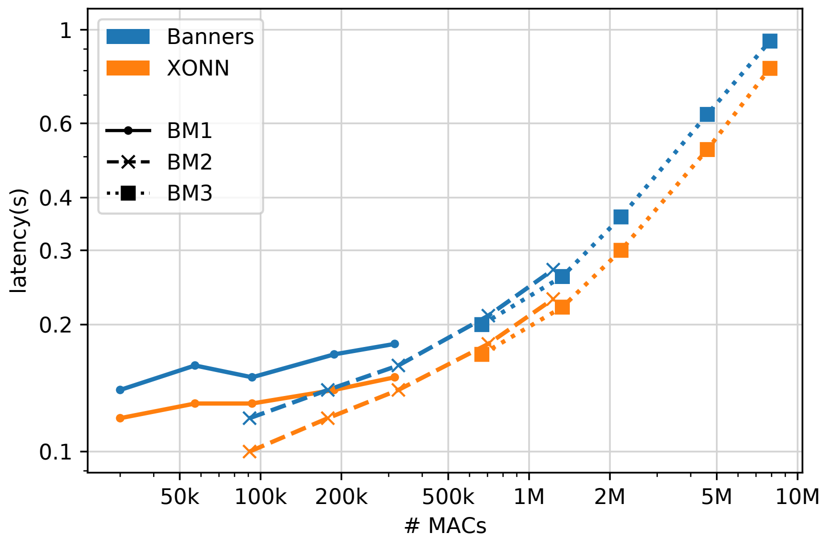


Figure 4.6: Tradeoff MACs - Latency for MNIST BNN

Table 4.2: Accuracy, communication, and latency comparisons[2] for MNIST dataset, Banners VS XONN [207].

Arch.	s	# param ($_{\times}10^3$)	# MACs ($_{\times}10^3$)	Accuracy (%)	Communication (MB)		Latency (s)		Est. RAM (MB)	
					Banners	XONN	Banners	XONN	Banners	XONN
	1	31	30	97.10	2.30	2.57	0.14	0.12	0.77	0.86
	1.5	58	57	97.56	3.53	4.09	0.16	0.13	1.18	1.36
BM1	2	94	93	97.82	5.13	5.87	0.15	0.13	1.71	1.96
	3	190	188	98.10	9.00	10.22	0.17	0.14	3.00	3.41
	4	320	316	98.34	13.73	15.62	0.18	0.15	4.58	5.21
	1	74	91	97.25	2.54	2.90	0.12	0.10	0.85	0.97
	1.50	153	178	97.93	5.03	5.55	0.14	0.12	1.68	1.85
BM2	2	291	326	98.28	9.14	10.09	0.16	0.14	3.05	3.36
	3	652	705	98.56	18.87	21.90	0.21	0.18	6.29	7.30
	4	1160	1230	98.64	33.42	38.30	0.27	0.23	11.14	12.77
	1	34	667	98.54	15.36	17.59	0.20	0.17	2.56	2.93
	1.5	75	1330	98.93	32.22	36.72	0.26	0.22	5.37	6.12
BM3	2	132	2200	99.13	56.35	62.77	0.36	0.3	9.39	10.46
	3	293	4610	99.26	117.11	135.88	0.63	0.52	19.52	22.65
	4	519	7890	99.35	207.40	236.78	0.94	0.81	34.57	39.46

[5]Note that the direct relation between MACs and complexity applies to sequential NN architectures like the ones described in this chapter. It does not hold for Recurrent Neural Networks and other non-sequential NN architectures.

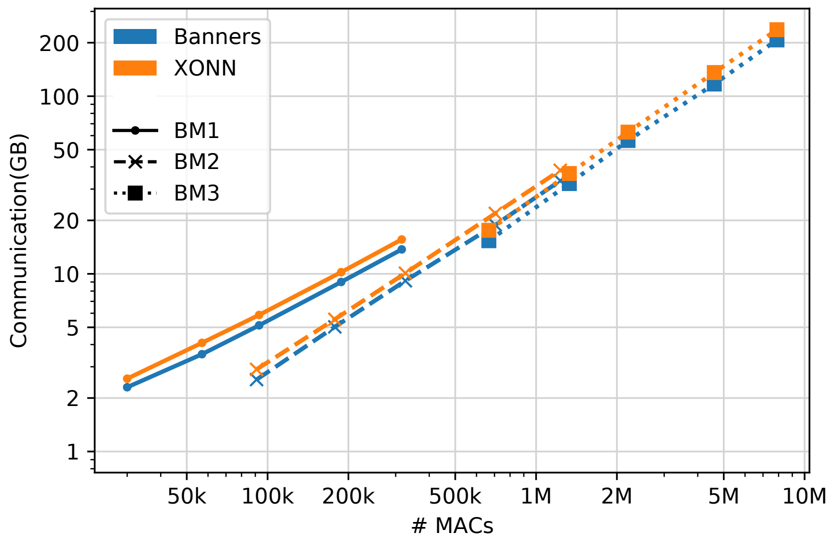


Figure 4.7: Tradeoff MACs - Communication for MNIST BNN

Finally, and crucial for edge devices, since all models are sequential, the RAM memory usage can be roughly estimated by the communication load divided by the number of layers (table 9 of [240]). By doing so we observe (last column in tables 4.1.5.2 and 4.1.5.2) that the memory footprint is small enough (tens to hundreds of MB) for Banners to be applicable for secure edge computing (e.g., a Raspberri Pi and most modern phones have 1GB+ of RAM).

4.1.6 Conclusion

With the formulation presented in this work, Banners aims to provide an efficient secure inference implementation of BNN by relying on Replicated Secret Sharing. All in all, the memory and space efficiency (R6), coupled with improved security protecting against one malicious adversary (CH3), provides a suitable candidate to run secure BNN inference on edge devices.

Table 4.3: Accuracy, communication, and latency comparisons[2] for CIFAR10 dataset, BANNERS VS XONN [207].

Arch.	s	# param (x10^3)	# MACs (x10^6)	Accuracy (%)	Communication (GB)		Latency (s)		Est. RAM (MB)	
					BANNERS	XONN [3]	BANNERS	XONN	BANNERS	XONN
BC1	1	200	42	72.0	1.22	1.26	5.02	3.96	122	126
	1.5	446	92	77.0	2.60	2.82	10.89	8.59	260	282
	2	788	163	80.0	4.79	4.98	18.90	15.07	479	498
	3	1760	364	83.0	10.31	11.15	43.16	33.49	1031	1115
BC2	1	92	12	67.0	0.37	0.39	1.77	1.37	28	30
	1.5	205	27	73.0	0.83	0.86	3.53	2.78	64	66
	2	363	47	78.0	1.48	1.53	6.08	4.75	114	118
	3	815	105	82.0	3.18	3.40	13.28	10.35	245	262
BC3	1	368	41	77.0	1.29	1.35	5.39	4.23	99	104
	1.5	824	92	81.0	2.84	3.00	11.52	9.17	218	231
	2	1460	164	83.0	4.97	5.32	20.72	16.09	382	409
	3	3290	369	86.0	11.03	11.89	45.67	35.77	848	915
BC4	1	689	143	82.0	4.36	4.66	17.78	14.12	291	311
	1.5	1550	322	85.0	9.88	10.41	39.98	31.33	659	694
	2	2750	572	87.0	17.87	18.45	69.36	55.38	1191	1230
	3	6170	1290	88.0	38.56	41.37	158.79	123.94	2571	2758
BC5	1	1210	166	81.0	5.26	5.54	21.17	16.78	250	264
	1.5	2710	372	85.0	11.68	12.40	46.78	37.29	556	590
	2	4810	661	86.0	20.51	21.98	83.75	65.94	977	1047
	3	10800	1490	88.0	46.04	49.30	190.14	147.66	2192	2348
BC6	1	1260	23	67.0	0.60	0.65	2.74	2.15	32	34
	1.5	2830	50	74.0	1.40	1.46	5.80	4.55	74	77
	2	5020	90	78.0	2.48	2.58	10.03	7.91	131	136
	3	11300	201	80.0	5.58	5.77	22.44	17.44	294	304

[2] The accuracy in BANNERS models matches the one described in this table by $\pm 0.1\%$. The number of parameters and number of Multiply-ACcumulate (MAC) are obtained from Larq. The communication and latency for XONN are taken from [207], while figures reported for BANNERS are yielded by MP-SPDZ.

Chapter 5

Protecting the Biometric Verification

The computation of privacy-preserving distance metrics $f_{dist}(\boldsymbol{x}, \boldsymbol{y})$ between two vectors $\boldsymbol{x}, \boldsymbol{y}$ followed by a comparison with a threshold θ is an essential block for biometric verification (both 1:1 authentication [165, 212] and 1:K identification [98]), while being very popular in many applications in need of privacy protection, including machine learning (e.g., k-nearest neighbors [250], linear regression [106], matrix multiplication [178]), bioinformatics (e.g., genetic relatedness [87]).

The literature counts many solutions based on various cryptographic techniques that allow distance metrics computation of over sensitive data while preserving its privacy: MPC-based solutions [88, 55, 99] split the distance computation across multiple entities, solutions rooted in FHE [60, 20, 27] resort to lattice-based public-key encryption schemes to support local addition and multiplication between ciphertexts, and solutions grounded on FE [19, 36] utilize public-key encryption schemes that support evaluation of scalar products when decrypting the ciphertexts.

However, not all operations are born equal. While linear operations are widely covered by all the privacy-preserving techniques, the protection of non-linear operations including the comparison to a threshold θ is much harder to attain. Computing this non-linear operation with most MPC primitives is often communication intensive (e.g., [88, 240]) both in terms of communication size and in number of rounds. FHE-based techniques must resort to computation-heavy algorithms [64, 138], and efficient FE-based techniques do not support non-linear function evaluations[1].

We devote this chapter to the study of privacy-preserving biometric verification. We seek to guarantee unlinkability (R3) for the biometric templates by ensuring their privacy (R1) throughout the entire process, while allowing the secure evaluation of both the linear (distance metric computation) and non-linear (comparison and $\arg\max$) functions that compose this phase. Additionally, we aim to maintain a high degree of accuracy (R5) in the

[1]While FE can, in theory, support arbitrary function evaluations, the instantiation of inner product and threshold comparison with FE would be extremely computationally intensive and thus completely impractical for real-world applications. This inefficiency is exacerbated by the comparative performance of MPC and FHE for the same tasks.

underlying biometric system, leading us to carefully study the various sources of accuracy losses (CH1).

This chapter is divided as follows. We present a general survey of the state of the art in this domain in Section 5.1, and follow-up with three novel contributions of this thesis in the next sections 5.2-5.4. Our main contributions are:

- BiomFEtrics [133] (Section 5.2), a FE-based protocol optimized for online latency that allows to securely compute the scalar product between templates (without comparison to a threshold), implemented and applied to real biometric data. We focus on the latency optimization (R6) and accuracy preservation (R5) on this chapter, leaving for Chapter 6 the study of the leakage in the output.

- Funshade [134] (Section 5.3), a new MPC protocol based on a smart combination of ΠSS and FSS to securely compute various distance metrics between templates with thresholding, intended for a 2PC semi-honest scenario.

- Grote [131] (Section 5.4), a novel CKKS-based algorithm to approximate the max of a set by applying group testing, a trick that is exploited to perform privacy-preserving face identification.

5.1 Existing Privacy-Preserving Biometric Solutions

The topic of protecting privacy in biometric verification systems has received a lot of attention, with considerable research focused on the protection of the templates (R3) and the secure verification (R2). This objective has been attained throughout a myriad of techniques [200]:

- *Fuzzy/robust hashing* [228, 218, 144, 229], where the templates are mapped via one-way functions to seemingly uncorrelated structures that tolerate a certain degree of noise/error when comparing them.

- *Cancellable biometrics* [217], where the templates transformed using geometric approaches (e.g. translations, rotations, affine transformations) to unlink them from the original biometric data, and operating the biometric matching on the transformed templates [232, 198, 71, 159]

- General privacy-preserving techniques such as *Homomorphic Encryption*, *Secure Multi-party Computation* and *Functional Encryption*, where the templates are protected via encryption/masking and the biometric operations are performed on the encrypted/-masked domain. A very promising set of solutions due to their flexibility, efficiency, and their provable security properties, these techniques drive the many of the recent progresses in the topic of secure biometrics, and thus they constitute the focus of this chapter.

In the domain of *Homomorphic Encryption* (HE), the biometrics use-case has led to a variety of approaches, including [20, 179] for Hamming distance or [245] for scalar product. However, these approaches do not include comparison to a threshold, and often rely on costly cryptographic primitives that make them slow. [28] and other homomorphic encryption based solutions such as [156] have gone the same way. [155] recently applied HE to iris recognition, while [118] employed HE for multi-biometric systems. A collection of HE-based solutions was surveyed in [212].

In a similar line, the *Multi-party Computation* (MPC) field includes a plethora of works covering distance metric evaluations. The privacy-preserving evaluation of a wide variety of distance metrics, resorting to Oblivious Transfer [199] and building on a previous work [45], was central to [44]. Most of the frameworks for privacy preserving neural networks cover scalar-products(e.g., [239, 158]). Mixed-mode protocols, using a combination of several MPC techniques or even HE, have also tackled secure distance evaluations [88, 185]. [99] covered a wide variety of Two-Party Computation techniques applied to biometrics.

Likewise, techniques based on *Functional Encryption* such as [4, 153] envisioned privacy-preserving biometric verification use-cases, employing the Hamming distance as their metric. There are other works in this line, such as [249] for biometric authentication using threshold predicate encryption, or [165] with its focus on high throughput.

Lastly, face biometrics has been subject of extensive studies before employing various privacy-preserving techniques: [215] using HE and MPC, [188] using Oblivious Transfer (OT), or more recently [189] using a mix of hashing and HE, and [92] employing multiple HE schemes for face identification.

5.2 BiomFEtrics: Practical Privacy-Preserving Face Identification with Function-Hiding Functional Encryption [133]

Abstract. Leveraging on function-hiding Functional Encryption (FE) and inner-product-based matching, this work presents a practical privacy-preserving face identification system with two key novelties: switching functionalities of encryption and key generation algorithms of FE to optimize matching latency (R6) while maintaining its security guarantees (CH2), and identifying leakage in the output (CH4) to later formalize two new attacks based on it with appropriate countermeasures[2]. We validate our scheme in a realistic face matching scenario, attesting its applicability to pseudo real-time one-use face identification scenarios like passenger identification.

5.2.1 Introduction

This work is our first iteration in the study on protection of the biometric verification operations. Introduced in Chapter 3, advanced cryptographic techniques such as Fully Homomorphic Encryption, Secure Multi-party Computation, and Functional Encryption

[2]We leave the study of this leakage for Chapter 6

can be used to address the limitations of standard cryptography (e.g., applicable mainly to secure storage and communication) and preserve the privacy of biometric data while in use.

To publicly decrypt/open the output of a given private function evaluation, MPC requires one round of communication. FHE requires the private key for its decryption, leading to extra security constraints required to protect this key. In comparison, FE is well suited for in-place function evaluation and output disclosure. Accordingly, in order to protect the biometric verification phase while addressing the need of local output decryption this work relies on Functional Encryption. FE is costly for arbitrary function evaluations [114], making it less practical for private evaluation of complex functions like those present in the live template acquisition step; nonetheless there exist efficient FE schemes to instantiate the inner product used during the biometric verification phase.

Our Contributions. This work presents a new face identification solution built on FE-based private inner product matching, with the key novelty of switching functionalities of encryption and key generation FE algorithms to optimize latency while leaving the strong security guarantees provided by FE intact. In addition, we perform a thorough a security analysis of the inner product input leakage, proposing countermeasures to thwart attacks based on it (Note that we leave this analysis for Chapter 6). We validate our solution in a face matching scenario, attesting its applicability practical one-use identification use-cases.

BiomFEtrics is outlined as follows. Section 5.2.2 describes the practical scenario and outlines our security goals for it. Section 5.2.3 details the proposed solution, architecture and characteristics. Section 5.2.4 comprises implementation and experiments. Section 5.2.5 addresses previous work and positions our contribution, wrapping up with the conclusions in Section 5.2.6. We leave the study of the inner product leakage (present in the original publication [133]) for Section 6.1.

5.2.2 A face identification scenario and its security goals

We consider the face identification scenario depicted in Section 2.1.2 and based on the blueprint of Figure 6.3, where the enrollment phase stores K reference templates in a privacy-preserving manner, and the verification phase computes similarity scores between the K reference templates and the live template, while preserving the privacy of all templates. In this work we employ the `FHIPE` scheme (Section 3.4.2) to provide the privacy-preserving property in our system.

This scenario leads to two practical considerations.

1. High numerical precision is paired with low error rates but FHIPE supports only integer operations (CH1), forcing us to deal with the conversion of floating-point biometric templates into integers.

2. We expect our end-to-end biometric identification to be performed in pseudo real time for it to be of practical use, hence we set an upper limit of up to 5s for its complete online execution. Since there are K identities in the DB, K similarity score computations are required for each verification, creating a natural bound to K to obtain an acceptable performance (R6).

We exemplify the applicability of this work in a use-case of identification for transport boarding (e.g., boarding a flight in an airport, or a bus/train in a station), requiring one-time-per-passenger identification of tens to low hundreds of individuals. A sketch of the system is shown in Fig. 5.1.

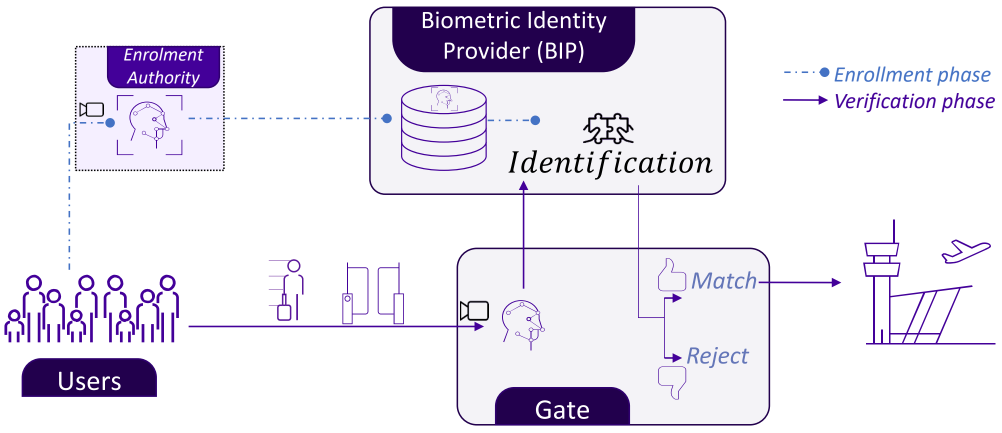


Figure 5.1: Sketch of the entities involved in a face identification scenario for one-time passenger access control.

Security goals. At this point we establish the security goals of our solution:

- **Privacy of all templates**. The enrollment phase should store *reference templates* in a privacy preserving manner still allowing inner products. Likewise, extracted *live templates* should support the inner product computation while remaining private for any other use. This is a standard by-design security goal of FE, already covered by the FHIPE scheme [153] of our solution.

- **Protection against inner product leakage**. FE schemes do not treat the inherent leakage of the reference template when computing several inner product operations with it. We formalize this leakage in Section 6.1, and develop two practical leakage-based attacks: *reference template extraction* and *brute-force impersonation.* Usually overlooked in the secure computation literature, we stress the importance of this leakage in our face identification scenario, where multiple inner products are computed over the same reference template. To protect against them, in Chapter 6 we establish a limit to the total number of identification requests in our solution.

Threat model. We consider a semi-honest adversary corrupting the similarity score operation and all steps after that, seeking to obtain as much information as possible from the inputs but preserving their integrity. We consider the adversary to have oracle access to the matching phase, thus being able to submit chosen live biometric samples. Our system is built with trust on the enrollment and the capture modules, for they receive the *msk* which can decrypt any ciphertext.

It is worth noting that deep fakes, morphing attacks and other attacks directed towards the feature extraction are left out of the scope, as they should be conveniently addressed in the feature extractor implementation.

5.2.3 Our solution

We display our solution in Fig. 5.2. In the *enrollment phase*, the enrollment module acts as trusted authority to generate msk & pp and protect N ref. templates by converting them into functional keys sk_i. msk is sent to the capture module, and all sk_i along with pp are sent to the verification module. The *verification phase* starts with the access control step. The capture module then gets a live template x and encrypts it into c using msk. Afterwards, the verification module takes sk_i and c, computes their privacy-preserving inner product $z_i = bmx \cdot bmy_i$, compares the highest score $max(z_i)$ to the threshold θ, and returns a match with the ID/index i of the highest score, or nothing if rejected.

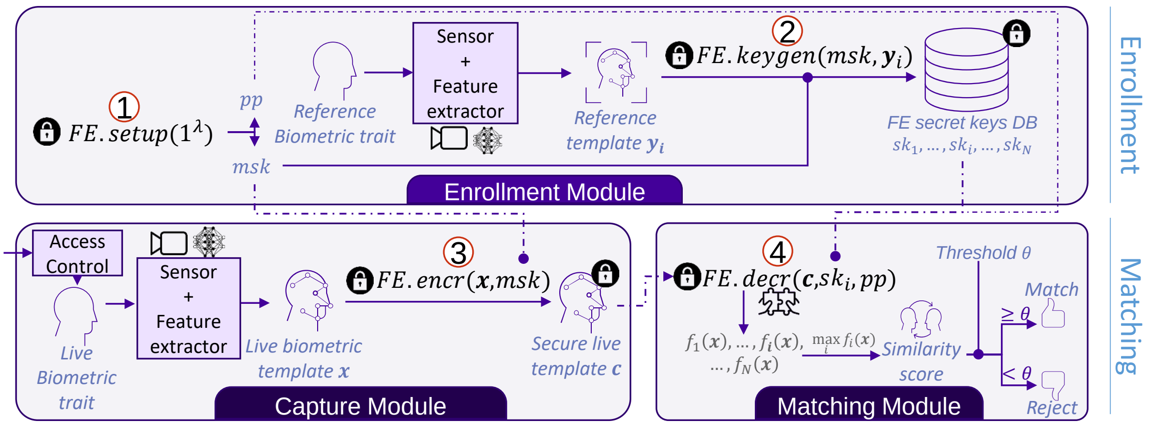


Figure 5.2: Architecture of our secure face identification system based on FE (FHIPE)

5.2.3.1 Swapping `FE.encr` with `FE.keygen`

The original FHIPE scheme (Sec. 5.1 of [153]) and posterior works based on it [142] use the function-hiding `FE.keygen` functionality to protect the live template (step 3 in Fig. 5.2), keeping `FE.encr` for the stored templates (step 2 in Fig. 5.2). We observe that, given the dual nature of the FHIPE scheme, the same security properties hold if we were to swap them. This observation is grounded on remark 3.4.5. of [36]: in the game-based IND-CPA security definition of FHIPE (Fig. 3.10 of [36] or definition 2.1) the adversary and the oracle follow a perfectly equivalent game. To optimize the end-to-end biometric verification latency we employ the fastest functionality for this phase, which happens to be `FE.encr` (see Sec. 5.2.4), thereby swapping `FE.encr` $\rightleftharpoons$ `FE.keygen` with respect to [153, 142].

5.2.3.2 Limiting the number of requests

As we will discuss in Section 6.1, we limit the number N of identification requests of our solution to prevent several attacks leveraging on the leakage of the output of `FE.decrypt` (wait for Section 6.1.1 for a detailed explanation and a careful selection of the limit). We enforce this limit via an *access control step* with open instantiation, which could materialize as an agent-controlled checkpoint or a one-time token generated in the enrollment.

5.2.3.3 Fixed-point approximation

The feature extractor outputs normalized templates $bmt \in \mathbb{R}^K_{[-1,1]}$, easily projected into the FHIPE discrete space $\mathbb{Z}_{2^l}$ by scaling with factor 2^l and a truncation to l bits. The subsequent inner product is naturally up-scaled twice:

$$f(bmx_{fix}, bmy_{fix}) = \left\lfloor bmx_{float} * 2^l \right\rfloor_l \cdot \left\lfloor bmy_{float} * 2^l \right\rfloor_l \approx 2^{2l} * f(bmx, bmy)$$

To compare against the threshold $\theta \in [0,1]$ we upscale θ twice: $\theta_{fix} = \theta * 2^{2l}$, obtaining an equivalent comparison. This fixed-point translation imposes a minimum ring size of $2l$ bits to avoid overflows. The approximation impacts the accuracy (CH1), since more bits yield more precision, but at the cost of bigger primitives in the FE scheme and thus worse latency. We study this trade-off in Sec. 5.2.4.

5.2.3.4 Security Analysis

We argue that BiomFEtrics protects the privacy of the underlying biometric system (R1).

Theorem 1. *Our system preserves the privacy of the live template and the reference templates while allowing the inner product similarity computation.*

Proof. The security of FE sits upon game-based definitions that prove Indistinguishability against Chosen Plaintext Attacks a.k.a. IND-CPA (*IND* [4, 34]). The FHIPE scheme of our solution is proven to hold strong SIM-based security guarantees as per theorem 3.1 of [153], which implies IND-CPA secure in Remark 2.5 of [153]. This directly ensures the privacy of the biometric templates inside ciphertexts and functional secret keys of our solution. □ □

5.2.4 Implementation & Experiments

We implement our Cython-based solution using the CiFEr [197] library, an ArcFace based [89] feature extractor with templates of size $K = 128$. The experiments were run in an Intel(R) Core(TM) i7-7800X CPU and averaged over 10 runs.

Table 5.1: Latency (seconds) for single-core FE.decr with template elements of l bits.

l	2	4	6	8	10	12	14	16
`FE.decrypt`	0.18	0.18	0.19	0.25	0.40	1.08	3.81	14.86

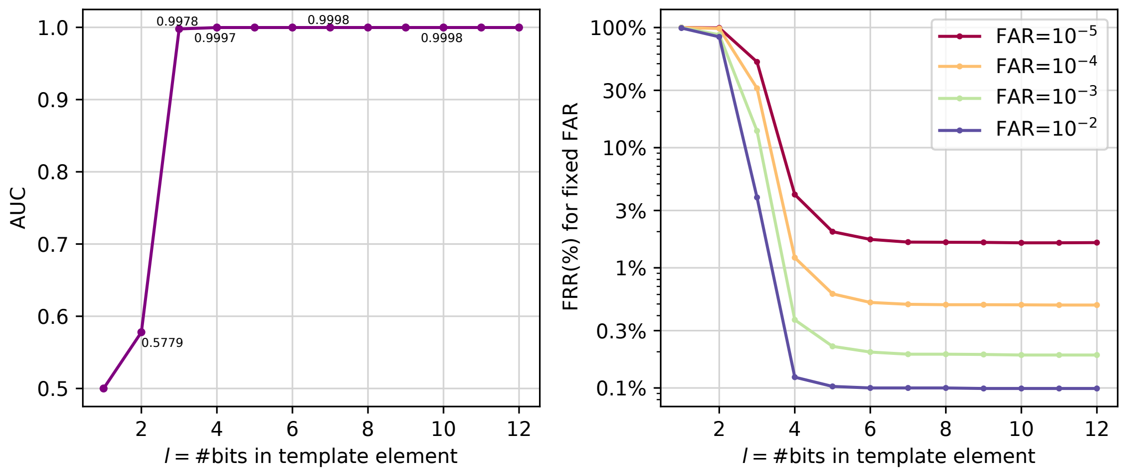


Figure 5.3: Comparison of the precision of the entire system for templates with elements of size l. In the left the Area Under the Curve metric is shown. In the right the False Rejection Rates for fixed False Acceptance Rates are displayed.

Latency. Optimizing the execution time of the verification phase is essential to make our system practical. Using a single core, we measure $FE.setup$ (step 1) to take 0.35s, `FE.keygen` (step 2) requires 0.19s per key, and `FE.encr` (step 3) demands 0.082s; thus our proposed swapping reduces the latency of live template protection by 55%. As the only FE operation depending on the template element size, $FE.decr$ latency is recorded in table 5.1. The feature extractor clocks 36 ± 1ms[3]. We disregard the latency of the access control step, as its instantiation is left open; and the max, the comparison with θ and the secure transmission for being negligible compared to the cost of FE operations.

Precision. We measure the accuracy of the system is measured with face identification benchmarks using the Labeled Faces in the Wild (LFW) dataset [127] consisting of 13233 112x112px real face images of famous people. We employ the FAR-FRR tradeoff as accuracy metric (Section 2.1.1). Typically, robust identification systems enforce $FAR < 10^{-3}$, obtaining a corresponding FRR. In Figure 5.3, we remark that highly compressed templates maintain high precision, with little improvement beyond $l = 6$.

To close up, Figure 5.4 presents the best trade-offs in two scenarios:

- **Higher precision**: Optimizing for low FRR, setting $l = 5$ bits per template element to support up to 70 identities, with slower matching of up to $5s$.
- **Many identities**, optimizing for high N (up to 100 identities) by setting $l = 4$ bits, at the cost of +2% FRR but with faster matching ($\approx 4s$).

[3]ArcFace-based [89] feature extractors with comparable latency and precision can be obtained from `https://github.com/deepinsight/insightface/wiki/Model-Zoo`

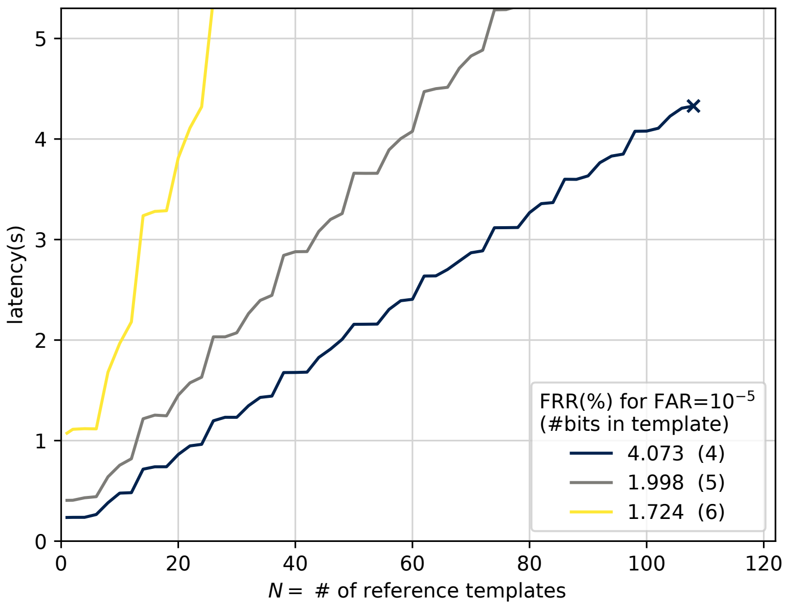


Figure 5.4: Practical choices of parameters for a realistic trade-off between matching latency, overall system precision (coupled to template element size l) and maximum number of identities N.

5.2.5 Related Works

The study of IPE started off with selective security in [4], already envisioning biometric use-cases, and reaching full security with [84, 234]. The function-hiding properties for IPE were introduced in [153], applied to biometric authentication based on Hamming weight ($l = 1$). Further works in function-hiding approaches include [154] and [142]. [19] covers an overhaul of efficient techniques.

The use of FE for privacy-preserving biometrics has also been subject to intense scrutiny, from [249] for biometric authentication using threshold predicate encryption, to the extreme efficiency of [165]. Whereas these works employ Hamming-weight based matchings that do not require approximations (typical from fingerprint or iris), our work tackles the cosine-similarity based matching of face biometrics. [7] covers an exhaustive revision of face recognition, which includes the LFW dataset [127] and the foundations of our feature extractor [89].

Among the most recent works, [142] proposes a useful acceleration trick for the FE scheme of [153], by caching all the repetitive computation depending only of the stored templates, obtaining up to 30% speedups. Much like the original [153], their function-hiding approach uses $FE.encr$ for the stored templates and $FE.keygen$ for the live templates. Comparatively, our function-hiding solution swaps $FE.encr \rightleftharpoons FE.keygen$ to optimize the latency of the system, and deals with the FHIPE leakage, often overlooked by the literature.

5.2.6 Conclusion

BiomFEtrics proposes an efficient, precise and privacy-preserving face identification system based on function-hiding functional encryption. We optimize the verification phase latency by swapping `FE.encr` and `FE.keygen` usage, speeding up the live template protection by 55% while maintaining the FE security guarantees. We also propose a fixed-point approximation to reduce the precision loss of the inner product, as well as a limit on the number of identification requests to prevent attacks grounded on the leakage in the output of our verification system. Finally, we implemented this system, showing that 4/5 bits per template element are enough to obtain precise setups that compute matchings against a database of up to 100 identities in pseudo real-time, applicable to passenger identification use-cases.

5.3 Funshade: Functional Secret Sharing for Two-Party Secure Thresholded Distance Evaluation [134]

Abstract. We propose a novel privacy-preserving, two-party computation of various distance metrics (e.g., Hamming distance, Scalar Product) followed by a comparison with a fixed threshold, which is known as one of the most useful and popular building blocks for many different applications including machine learning, biometric matching, etc. Our solution builds upon recent advances in Functional Secret Sharing and makes use of an optimized version of arithmetic secret sharing dubbed ΠSS (Section 3.2.3). Thanks to this combination, our new solution named Funshade is the first to require only one round of communication and two ring elements of communication in the online phase, outperforming all prior state-of-the-art schemes while relying on lightweight cryptographic primitives. Lastly, we implement the solution from scratch in Python using efficient C++ blocks, testifying its high performance.

5.3.1 Introduction

We shift from FE to MPC, with the objective of also covering the comparison to θ in the matching. When used to evaluate circuits based on only binary or only arithmetic interactions, MPC protocols present very fast online execution. However, applications such as biometrics or machine learning require a combination of linear operations (additions and multiplications over a large ring) and non-linear operations such as integer comparison or truncation. The cost of blindly implementing these two types of operations with only one MPC circuit type can be significantly high. To address this, many works have tackled mixed-mode MPC to provide efficient conversions between arithmetic and binary domains, supporting both linear and non-linear operations [55, 88, 185, 193]. Yet, these conversions often entail a hefty communication overhead in the online phase both in terms of size and number of rounds.

Inspired by the TinyTable protocol [80] to secret share truth tables in a succinct manner, Boyle et al. proposed a very promising approach [41, 38] based on FSS [39, 40]. Offering the same online communication and round complexity for non-linear function evaluations as for pure arithmetic computations in arithmetic-only circuits, FSS relies on fast symmetric cryptography primitives to yield a fast online evaluation.

The present work will benefit from $\Pi - SS$ (Section 3.2.3), an evolved secret sharing technique emanating from research in mixed-mode operations [193] and modern FSS-based comparison protocols [38] to achieve a lightweight and highly efficient biometric matching protocol.

Our Contributions. We draw inspiration from the family of distance metrics covered in GSHADE and integrate FSS-based threshold comparison primitives from [38] with an optimized version Secret Sharing [193] in a two-party computation (2PC) protocol to perform privacy-preserving distance metric computations with a subsequent comparison to θ. To summarize our contributions, our solution:

- requires just one round of communication in the online phase, lowering the communication costs with respect to the two-round state-of-the-art solutions from AriaNN [213] and Boyle et. al. [38],
- sends only two ring elements in the online phase, reducing the communication size of previous solutions by a factor of $2l$ (l being the size of the input vectors),
- features 100% correctness in the comparison result,
- is implemented and open-sourced in a standalone Python library with efficient C++ primitives.

Funshade is outlined as follows: Section 5.3.2 describes the distance metrics we consider in this work and some applications. Section 3 details the proposed solution, including a succinct security analysis. Section 4 addresses previous work and positions our contribution, wrapping up with the conclusions and next steps in Section 5.

5.3.2 Distance metrics and applications

We start off by writing the generic function we wish to protect:

$$f(f_{dist}, \theta, \boldsymbol{x}, \boldsymbol{y}) = 1_{f_{dist}(\boldsymbol{x},\boldsymbol{y}) \geqslant \theta} = \begin{cases} 1 & \text{if } f_{dist}(\boldsymbol{x}, \boldsymbol{y}) \geqslant \theta, \\ 0 & \text{if } f_{dist}(\boldsymbol{x}, \boldsymbol{y}) < \theta, \end{cases} \tag{5.1}$$

Inspired by GSHADE [44], and extending the list of distance metrics from Section 2.1, we introduce below the distance metrics f_{dist} that we cover in Funshade alongside motivating real-world applications:

- **Scalar Product**: $f_{SP}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{y} = \sum_{i=1}^{n} \boldsymbol{x}^{(i)} \boldsymbol{y}^{(i)}$ is a common distance metric in face recognition where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ are two vectors of the same dimension.
- **Hamming Distance**: $f_{HD}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} (\boldsymbol{x}^{(i)} \oplus \boldsymbol{y}^{(i)})$ is a distance metric frequently used in information theory and computer science to measure the distance between two bit-strings. Besides its interest in iris and fingerprint recognition, it is the base of the perceptual hashing technique [183] used in image comparison, with applications

ranging from image watermarking [94] to detection of Child Sexual Abuse Material (CSAM) [76].

- **Squared Euclidean Distance**: $f_{SED}(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{n}(\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)})^2$ is a distance metric used in many machine learning applications, such as clustering [181]. It is also used in the context of face recognition [111].

- **Squared Mahalanobis Distance**: $f_{MD}(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T\boldsymbol{M}\ (\boldsymbol{x}-\boldsymbol{y})$ is a distance metric used in many machine learning applications, such as clustering [181] and recognition of hand shape/keystrokes/signatures [44].

To adapt to 2PC, we reformulate these distance metrics f_{dist} as

$$z = f_{dist}(\boldsymbol{x},\boldsymbol{y}) = f_{local}(\boldsymbol{x}) + f_{local}(\boldsymbol{y}) + f_{cp}\cdot\sum(\boldsymbol{x}^{(i)}\cdot\boldsymbol{y}^{(i)}) \tag{5.2}$$

where f_{local} is a function that can be computed locally by each input data holder, and f_{cp} is the "cross product" constant factor that applies to the scalar product evaluation present in all the metrics. Using this blueprint, we rewrite all the distance metrics in Table 5.2.

We remark that the Hamming Distance can be reformulated as the Squared Euclidean Distance as long as the input vectors are composed of binary values $\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}\in\{0,1\}\forall i$, since the boolean XOR operation between two binary values can be rewritten in the arithmetic domain as $\boldsymbol{x}^{(i)}\oplus\boldsymbol{y}^{(i)} = (\boldsymbol{x}^{(i)}-\boldsymbol{y}^{(i)})^2$, the square of its difference.

Table 5.2: Reformulation of the distance metrics into a composition of local evaluations of f_{local} and the cross product $f_{cp}\cdot\boldsymbol{x}^T\boldsymbol{y}$

Distance Metric	Formula	$f_{local}(\boldsymbol{x}) + f_{local}(\boldsymbol{y})\ + f_{cp}\cdot\boldsymbol{x}^T\boldsymbol{y}$	$f_{local}(\boldsymbol{v})$	f_{cp}
Scalar/Inner Product	$\sum\boldsymbol{x}^{(i)}\cdot\boldsymbol{y}^{(i)}$	$0\ +\ 0\ +1\sum(\boldsymbol{x}^{(i)}\cdot\boldsymbol{y}^{(i)})$	0	1
Hamming Distance	$\sum\boldsymbol{x}^{(i)}\oplus\boldsymbol{y}^{(i)}$	$\sum(\boldsymbol{x}^{(i)})^2+\sum(\boldsymbol{y}^{(i)})^2-2\sum(\boldsymbol{x}^{(i)}\cdot\boldsymbol{y}^{(i)})$	$\sum(\boldsymbol{v})^2$	-2
Squared Euclidean	$\sum(\boldsymbol{x}^{(i)}-\boldsymbol{y}^{(i)})^2$	$\sum(\boldsymbol{x}^{(i)})^2+\sum(\boldsymbol{y}^{(i)})^2-2\sum(\boldsymbol{x}^{(i)}\cdot\boldsymbol{y}^{(i)})$	$\sum(\boldsymbol{v})^2$	-2
Squared Mahalanobis	$(\boldsymbol{x}-\boldsymbol{y})^T\boldsymbol{M}(\boldsymbol{x}-\boldsymbol{y})$	$\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x}\ +\ \boldsymbol{y}^T\boldsymbol{M}\boldsymbol{y}\ -2\,(\boldsymbol{x}^T\boldsymbol{M})\cdot\boldsymbol{y}$	$(\boldsymbol{v}^TM\boldsymbol{v})$	-2

5.3.3 Our solution

We now describe our solution for a lightweight and efficient 2PC distance metric with comparison, with a single round of communication in the online phase. In a nutshell, we combine Π-sharing to locally compute a scalar product with the FSS gate for interval containment from [38] with full correctness.

On the threat model and the security guarantees. FUNSHADE focuses on a 2PC scenario, following the party conventions and roles from 3.2. We cover security against a semi-honest adversary non-adaptively corrupting at most one computing party. Also referred to as *Honest -but-Curious*, the computing parties P_j are to follow the protocol faithfully, while a party corrupted by the adversary will try to extract as much information as possible from his computation.

Employing simulation based security proofs [48, 113], previous works have proven SS and ΠSS to be perfectly information theoretic secure against computationally unbounded semi-honest adversaries [88, 193] (*perferct hiding*). In contrast, FSS schemes FSS schemes rely on the security of the underlying PRNG to prove computational security against PPT bounded adversaries [38] (*computational hiding*).

5.3.3.1 Sketching the protocol

The key insight driving our design stems from the intermediate SS state in the Π-sharing multiplication ($\langle \Delta_z \rangle$ in Equation 3.3). By providing Π-shared input vectors to the computing parties P_j, we can locally obtain the SS shares of the elementwise multiplication, and perform local cumulative addition to obtain shares of the scalar product result. Compared to the pure SS approach, we no longer need a round of communication to reconstruct the intermediate values $x-a$ and $y-b$ masked by beaver triples (`SS.mult` in Equation 3.1). As pointed out in ABY2.0 [193], the communication in a ΠSS multiplication gate happens at the output wires, as opposed to SS multiplication gates where the round of communication is tied to the input wires.

The subsequent FSS gate for interval containment (`FSS.keygen` and `FSS.eval`, Algorithms 1 and 2) requires a publicly reconstructed input held by both parties. To preserve the input data privacy (R1), this input must be masked prior to its reconstruction (in line with previous FSS-based works [41, 38, 213]). Crucially, the masking of the private input via local shares addition followed by its reconstruction (at the cost of one round of communication) happens at the input wire of the FSS gate.

All we have left is to put together the two pieces of the puzzle. We can skip the Π-sharing reconstruction and instead add the input mask directly to the scalar product output, and then reconstruct this masked value to serve as public input for the FSS interval containment gate. Figure 5.5 depicts our idea applied to the scalar product metric.

To obtain the other metrics we would have each input data holder $\mathsf{P}_{in_x}, \mathsf{P}_{in_y}$ run f_{local} on its inputs and secret share the result with the computing parties to add it to the output of the scalar product. In addition to that, both parties would multiply the shares of the scalar product result with the corresponding f_{cp}, resulting in the correct distance metric evaluation $z = f_{dist}(\boldsymbol{x}, \boldsymbol{y})$.

To keep the threshold θ hidden from the computing parties (and known only by P_{setup}), we subtract the value of θ from the additive random mask r during the offline/setup phase, employing an IC gate (Protocols 1 and 2) and then compute the of $z_\theta = z - \theta$.

5.3.3.2 Protocol specification

Embracing this combination of ΠSS for the locally computed scalar product and FSS for the comparison to θ, we can now outline each of the protocols that compose the full solution.:

1. FUNSHADE.Setup (Protocol 11): P_{setup} generates the correlated randomness required for the scalar product multiplications, as well as the keys for the interval containment, and distributes the preprocessing material to the parties involved in the online phase.

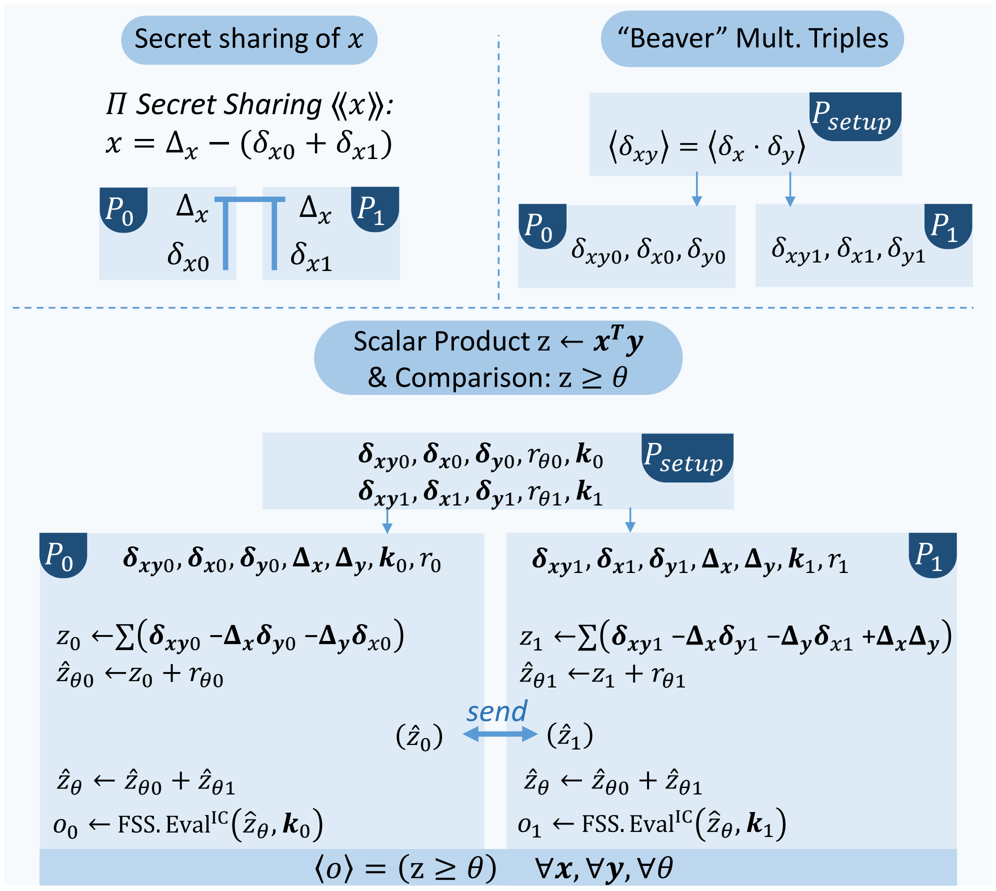

Secret sharing of x
Π Secret Sharing $\langle\langle x\rangle\rangle$:
$x = \Delta_x - (\delta_{x0} + \delta_{x1})$
P_0 Δ_x δ_{x0}
P_1 Δ_x δ_{x1}
"Beaver" Mult. Triples
P_{setup}
$\langle\delta_{xy}\rangle = \langle\delta_x \cdot \delta_y\rangle$
P_0 $\delta_{xy0}, \delta_{x0}, \delta_{y0}$
P_1 $\delta_{xy1}, \delta_{x1}, \delta_{y1}$
Scalar Product z ← $\boldsymbol{x}^T\boldsymbol{y}$
& Comparison: z ≥ θ
$\boldsymbol{\delta}_{xy0}, \boldsymbol{\delta}_{x0}, \boldsymbol{\delta}_{y0}, r_{\theta 0}, \boldsymbol{k}_0$
$\boldsymbol{\delta}_{xy1}, \boldsymbol{\delta}_{x1}, \boldsymbol{\delta}_{y1}, r_{\theta 1}, \boldsymbol{k}_1$
P_{setup}
P_0 $\boldsymbol{\delta}_{xy0}, \boldsymbol{\delta}_{x0}, \boldsymbol{\delta}_{y0}, \boldsymbol{\Delta}_x, \boldsymbol{\Delta}_y, \boldsymbol{k}_0, r_0$
$z_0 \leftarrow \sum(\boldsymbol{\delta}_{xy0} - \boldsymbol{\Delta}_x\boldsymbol{\delta}_{y0} - \boldsymbol{\Delta}_y\boldsymbol{\delta}_{x0})$
$\hat{z}_{\theta 0} \leftarrow z_0 + r_{\theta 0}$
$(\hat{z}_0)$
send
$(\hat{z}_1)$
$\hat{z}_\theta \leftarrow \hat{z}_{\theta 0} + \hat{z}_{\theta 1}$
$o_0 \leftarrow \mathrm{FSS.Eval}^{\mathrm{IC}}(\hat{z}_\theta, \boldsymbol{k}_0)$
P_1 $\boldsymbol{\delta}_{xy1}, \boldsymbol{\delta}_{x1}, \boldsymbol{\delta}_{y1}, \boldsymbol{\Delta}_x, \boldsymbol{\Delta}_y, \boldsymbol{k}_1, r_1$
$z_1 \leftarrow \sum(\boldsymbol{\delta}_{xy1} - \boldsymbol{\Delta}_x\boldsymbol{\delta}_{y1} - \boldsymbol{\Delta}_y\boldsymbol{\delta}_{x1} + \boldsymbol{\Delta}_x\boldsymbol{\Delta}_y)$
$\hat{z}_{\theta 1} \leftarrow z_1 + r_{\theta 1}$
$\hat{z}_\theta \leftarrow \hat{z}_{\theta 0} + \hat{z}_{\theta 1}$
$o_1 \leftarrow \mathrm{FSS.Eval}^{\mathrm{IC}}(\hat{z}_\theta, \boldsymbol{k}_1)$
$\langle o\rangle = (z \geq \theta) \quad \forall \boldsymbol{x}, \forall \boldsymbol{y}, \forall\theta$

Figure 5.5: Overview of Funshade primitives

2. Funshade.Share (Protocol 12): $\mathsf{P}_{in_x}, \mathsf{P}_{in_y}$, the input holder players, prepare the Π-shares of their corresponding inputs using the correlated randomness and then send these shares to the computing parties $\mathsf{P}_0, \mathsf{P}_1$.

3. Funshade.Eval (Protocol 13): $\mathsf{P}_0, \mathsf{P}_1$ engage in an online protocol upon acquiring the Π-shares of both inputs, using local multiplication and addition to compute the scalar product, and then evaluate the interval containment FSS scheme to determine whether the result is below the threshold θ.

4. Funshade.Result (Protocol 14): $\mathsf{P}_0, \mathsf{P}_1$ send the arithmetic shares of the result to the player designed to receive the output P_{res} for its reconstruction.

5.3.3.3 Applications and Practical considerations

We display a diagram of our solution applied to biometrics/CSAM detection in Figure 5.6. The Funshade protocol can be easily computed in parallel for different inputs $\boldsymbol{y}$ in cases where the reference database contains more than one record, such as CSAM detection against a large database of hashes or biometric identification against multiple subjects. Additionally, these use-cases normally gather their reference databases ahead of time. To

Protocol 11 $\textsc{Funshade}.\text{Setup}(l,\ n,\ \lambda,\ \theta) \to \boldsymbol{k}_0, \boldsymbol{k}_1, \langle \delta_{\boldsymbol{x}} \rangle, \langle \delta_{\boldsymbol{y}} \rangle$

Players: P_{setup} carries out all the setup.
Input: l: length of the input vectors.
n: number of bits for the secret sharing ring $\mathbb{Z}_{2^n}$.
λ: security parameter.
θ: threshold for the comparison $\in \mathbb{Z}_{2^n}$.
Output: $\boldsymbol{k}_0, \boldsymbol{k}_1$: preprocessing keys, sent to P_0, P_1 respectively.
$\langle \delta_{\boldsymbol{x}} \rangle, \langle \delta_{\boldsymbol{y}} \rangle$: δ-shares of input vectors, sent to $\mathsf{P}_{in_{\boldsymbol{x}}}, \mathsf{P}_{in_{\boldsymbol{y}}}$ (input owners) resp.
Note: All arithmetic operations ($+$,$-$,$\cdot$) are defined in $\mathbb{Z}_{2^n}$.

```
Beaver Triples for Π-sharing scalar product:
1: $\langle \boldsymbol{\delta}_x \rangle, \langle \boldsymbol{\delta}_y \rangle \equiv ((\boldsymbol{\delta}_{x_0}, \boldsymbol{\delta}_{x_1}),\ (\boldsymbol{\delta}_{y_0}, \boldsymbol{\delta}_{y_1})) \sim \mathcal{U}_{[\mathbb{Z}_{2^n}^{l\times 4}]}$
2: $\boldsymbol{\delta}_{xy_0} \sim \mathcal{U}_{[\mathbb{Z}_{2^n}^{l}]}$
   $\boldsymbol{\delta}_{xy_1} \leftarrow (\boldsymbol{\delta}_{x_0} + \boldsymbol{\delta}_{x_1}) \cdot (\boldsymbol{\delta}_{y_0} + \boldsymbol{\delta}_{y_1}) - \boldsymbol{\delta}_{xy_0}$
   $\langle \boldsymbol{\delta}_{xy} \rangle \equiv (\boldsymbol{\delta}_{xy_0}, \boldsymbol{\delta}_{xy_1})$
3: $\langle r \rangle \equiv (r_0, r_1) \sim \mathcal{U}_{[\mathbb{Z}_{2^n}^{\times 2}]}$        $r \leftarrow r_0 + r_1$
   $\langle r_\theta \rangle \equiv (r_{\theta 0}, r_{\theta 1}) \leftarrow (r_0, r_1 - \theta)$
FSS interval containment:
4: $\boldsymbol{k}_0^{IC}, \boldsymbol{k}_1^{IC} \leftarrow \text{FSS.Gen}^{IC}(\lambda, n, r)$
5: $\boldsymbol{k}_j \equiv (\boldsymbol{\delta}_{x_j}, \boldsymbol{\delta}_{y_j}, \boldsymbol{\delta}_{xy_j}, r_{\theta j}, \boldsymbol{k}_j^{IC}),\ j \in \{0, 1\}$
Dealing the preprocessing material :
6: Send $\boldsymbol{k}_0 \Rightarrow \mathsf{P}_0$,     $(\boldsymbol{\delta}_{\boldsymbol{x}_0}, \boldsymbol{\delta}_{\boldsymbol{x}_1}) \Rightarrow \mathsf{P}_{in_{\boldsymbol{x}}}$
        $\boldsymbol{k}_1 \Rightarrow \mathsf{P}_1$,     $(\boldsymbol{\delta}_{\boldsymbol{y}_0}, \boldsymbol{\delta}_{\boldsymbol{y}_1}) \Rightarrow \mathsf{P}_{in_{\boldsymbol{y}}}$
```

speed up the online phase, the reference database held by party $\mathsf{P}_{in_{\boldsymbol{y}}}$ could be Π-shared as part of the offline phase, leaving only the live input to be shared in the online phase. In addition, biometric identifications / CSAM detections might output one single bit to determine whether there is a match in the entire database. In this case the individual secret shared outputs $\boldsymbol{o}_j^{(i)}$ could be locally summed up to yield a single number as output.

As an alternative to the trusted setup carried by P_{setup}, the two computing parties $\mathsf{P}_0, \mathsf{P}_1$ could follow an interactive protocol in the offline phase to jointly realize the role of P_{setup} (execution of Funshade.Setup and distribution of key material), resorting to distributed generation via generic 2PC techniques for the FSS gate key generation (Appendix A.2 of [38]), and either Oblivious Transfer or Homomorphic Encryption for the ΠSS scalar product preprocessing material (Section 3.1.3 of [193]). [4]

5.3.4 Security analysis

We consider security against a Honest-but-Curious adversary $\mathcal{A}$ that corrupts up to one of the two computing parties P_j. We consider a static corruption model where the adversary must choose which participant to corrupt before the execution of the computations. This is a

[4]That being said, the trusted setup might be justified in a context of biometrics/CSAM detection. Not trusting the reference database would immediately defeat the purpose of the system. Hence, the system must trust the entity in possession of the reference database (e.g., $\mathsf{P}_{in_{\boldsymbol{Y}}}$), and thus this entity could naturally play the role of P_{setup}.

Protocol 12 FUNSHADE.Share($\boldsymbol{v}$, $\boldsymbol{\delta}_{\boldsymbol{v}0}$, $\boldsymbol{\delta}_{\boldsymbol{v}1}$) $\rightarrow \boldsymbol{\Delta}_v, \langle d_{\boldsymbol{v}} \rangle$

Players: P_{in_v}, holding the input vector $\boldsymbol{v}$ (where $\boldsymbol{v} \in \{\boldsymbol{x}, \boldsymbol{y}\}$).

Input: $\boldsymbol{v}$: input vector $\in \mathbb{Z}_{2^n}^l$ held by P_{in_v}.
$\boldsymbol{\delta}_{\boldsymbol{v}j}$: Precomputed δ-shares $\in \mathbb{Z}_{2^n}^l$.

Output: $\boldsymbol{\Delta}_v$: Δ-shares of vector $\boldsymbol{v}$ distributed to both P_0 & P_1.
$d_{\boldsymbol{v}_j}$: Arithmetic shares of the local computation $f_{local}(\boldsymbol{v})$.

```
1: $\boldsymbol{\Delta}_v \leftarrow (\boldsymbol{v} + \boldsymbol{\delta}_{\boldsymbol{v}0} + \boldsymbol{\delta}_{\boldsymbol{v}1})$
2: $d_{\boldsymbol{v}} \leftarrow f_{local}(\boldsymbol{v})$;   $\langle d_{\boldsymbol{v}} \rangle \equiv (d_{\boldsymbol{v}0}, d_{\boldsymbol{v}1}) \leftarrow (\sim \mathcal{U}_{[\mathbb{Z}_{2^n}]}, d_{\boldsymbol{v}} - d_{\boldsymbol{v}0})$
3: SEND $(\boldsymbol{\Delta}_v, d_{\boldsymbol{v}_0}) \Rightarrow \mathsf{P}_0$,   $(\boldsymbol{\Delta}_v, d_{\boldsymbol{v}_1}) \Rightarrow \mathsf{P}_1$
```

Protocol 13 FUNSHADE.Eval($j, \Delta_{\boldsymbol{x}}, \Delta_{\boldsymbol{y}}, \langle d_{\boldsymbol{x}} \rangle, \langle d_{\boldsymbol{y}} \rangle, \boldsymbol{k}_j$) $\rightarrow \langle o \rangle$

Players: P_j, $j \in \{0, 1\}$ computing parties.

Input: $\Delta_{\boldsymbol{x}}, \Delta_{\boldsymbol{y}}$: Δ-shares of $⟪\boldsymbol{x}⟫, ⟪\boldsymbol{y}⟫$ (Π-shared inputs $\boldsymbol{x}, \boldsymbol{y}$) held by both P_0 and P_1.
$\langle d_{\boldsymbol{x}} \rangle, \langle d_{\boldsymbol{y}} \rangle$: Arithmetic shares of locally computed single-input terms $f_{local}(\boldsymbol{x}), f_{local}(\boldsymbol{y})$ of $f_{dist}(\boldsymbol{x}, \boldsymbol{y})$.
$\boldsymbol{k}_j$: preprocessing keys from FUNSHADE.Setup containing:
$\boldsymbol{\delta}_{\boldsymbol{x}_j}, \boldsymbol{\delta}_{\boldsymbol{y}_j}$: δ-shares of Π-shared input vectors $\boldsymbol{x}, \boldsymbol{y}$,
$\boldsymbol{\delta}_{\boldsymbol{xy}_j}$: arith. shares of Beaver triple s.t. $\langle \boldsymbol{\delta}_x \rangle \langle \boldsymbol{\delta}_y \rangle = \langle \boldsymbol{\delta}_{xy} \rangle$,
$r_{\theta j}$: arith. shares of FSS input mask r minus threshold θ,
$\boldsymbol{k}_j^{IC}$: FSS key for the IC gate of [38].

Output: $\langle o \rangle$: arithmetic shares of the result $o = f(\boldsymbol{x}, \boldsymbol{y}) \geq \theta$.

Note: All steps apply to both computing parties P_j, $j \in \{0, 1\}$. All arithmetic operations $(+, -, \cdot)$ are defined in $\mathbb{Z}_{2^n}$.

```
$\Pi$-sharing based scalar product:
1: $\hat{z}_{\theta j} \leftarrow r_{\theta j} + d_{\boldsymbol{x}_j} + d_{\boldsymbol{y}_j} + f_{cpf} \cdot \sum^l [j \cdot \Delta_{\boldsymbol{x}} \cdot \Delta_{\boldsymbol{y}} - \Delta_{\boldsymbol{x}} \cdot \delta_{\boldsymbol{y}_j} - \Delta_{\boldsymbol{y}} \cdot \delta_{\boldsymbol{x}_j} + \delta_{\boldsymbol{xy}_j}]$
Reconstruction of masked input to FSS gate:
2: $\mathsf{P}_j$: SEND $\hat{z}_{\theta j} \Rightarrow \mathsf{P}_{1-j}$;   $\hat{z}_\theta \leftarrow \hat{z}_{\theta 0} + \hat{z}_{\theta 1}$
Interval Containment for sign extraction:
3: $o_j \leftarrow$ FSS.Eval$^{IC}(j, \boldsymbol{k}_j^{IC}, \hat{z}_\theta)$
4: return $o_j$
```

standard security model in previous MPC frameworks [38, 213, 193, 88, 185, 55]. Under this threat model, we define and later prove the security and correctness of our constructions.

We employ the standard real world - ideal world paradigm, providing the simulation for the case of a corrupt P_j. The ideal world simulation contains an additional trusted party that receives all the inputs from $\mathsf{P}_0, \mathsf{P}_1$, computes the ideal functionality correctly and sends the corresponding results back to $\mathsf{P}_0, \mathsf{P}_1$. Conversely, the real world simulation executes the protocol as described in the FUNSHADE algorithms in the presence of $\mathcal{A}$.

Our security proof works in the $\mathcal{F}_{\text{FUNSHADE}.setup}$-hybrid model where $\mathcal{F}_{\text{FUNSHADE}.setup}$ represents the ideal functionality corresponding to protocol FUNSHADE.setup.

Definition 3 (Security of Funshade). *For each $j \in \{0, 1\}$, there is a PPT algorithm $\mathcal{S}$ (simulator) such that $\forall \theta \in \mathbb{Z}_{n+}^*$, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_n^l$ and every function $f_{dist}(\boldsymbol{x}, \boldsymbol{y}) : \mathbb{Z}_n^l \rightarrow \mathbb{Z}_n$ from Table 5.2, $\mathcal{S}$ realizes the ideal functionality $\mathcal{F}_{th-dist}$, such that its behavior is computationally*

Protocol 14 FUNSHADE.Result($\langle o \rangle$) $\rightarrow o$

Players: P_j, $j \in \{0,1\}$ computing parties, P_{res} result holder.
Input: $\langle o \rangle$: secret shares $o_0, o_1 \in \mathbb{Z}_{2^n}$ of the result o held by $\mathsf{P}_0, \mathsf{P}_1$.
Output: o: Output value.

```
1: $\mathsf{P}_j$: SEND $o_j \Rightarrow \mathsf{P}_{res}$.
2: $\mathsf{P}_{res}$: $o \leftarrow (o_0 + o_1)$
```

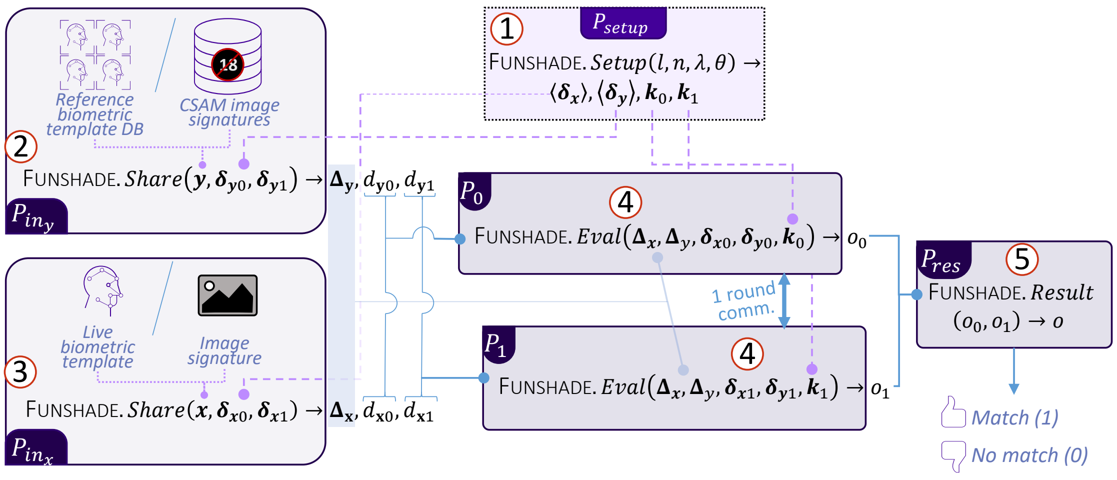

1
P_{setup}
FUNSHADE. Setup$(l, n, \lambda, \theta) \rightarrow \langle \delta_x \rangle, \langle \delta_y \rangle, k_0, k_1$
18
Reference biometric template DB
CSAM image signatures
2
FUNSHADE. Share$(y, \delta_{y0}, \delta_{y1}) \rightarrow \Delta_y, d_{y0}, d_{y1}$
P_{in_y}
Live biometric template
Image signature
3
FUNSHADE. Share$(x, \delta_{x0}, \delta_{x1}) \rightarrow \Delta_x, d_{x0}, d_{x1}$
P_{in_x}
P_0
4
FUNSHADE. Eval$(\Delta_x, \Delta_y, \delta_{x0}, \delta_{y0}, k_0) \rightarrow o_0$
1 round comm.
P_1
4
FUNSHADE. Eval$(\Delta_x, \Delta_y, \delta_{x1}, \delta_{y1}, k_1) \rightarrow o_1$
P_{res}
5
FUNSHADE. Result $(o_0, o_1) \rightarrow o$
Match (1)
No match (0)

Figure 5.6: Diagram of our Funshade protocol applied to biometrics/CSAM detection

indistinguishable from a real world execution of protocols 12-13-14 in the presence of a semi-honest adversary $\mathcal{A}$.

Ideal Functionality $\mathcal{F}_{th-dist}$
$\mathcal{F}_{th-dist}$ interacts with the parties $\mathsf{P}_0, \mathsf{P}_1$ and the adversary $\mathcal{S}$ and is parametrized by a publicly know function $f_{dist}(\boldsymbol{x}, \boldsymbol{y})$ and a threshold θ.

- ***Inputs***: $\mathcal{F}_{th-dist}$ receives the inputs $\Delta_{\boldsymbol{x}}, \Delta_{\boldsymbol{y}}, \delta_{\boldsymbol{x}_j}, \delta_{\boldsymbol{y}_j}$ from the computing parties $\mathsf{P}_0, \mathsf{P}_1$.
- ***Computation***: $\mathcal{F}_{th-dist}$ reconstructs $\boldsymbol{x} = \Delta_{\boldsymbol{x}} - (\delta_{\boldsymbol{x}_0} + \delta_{\boldsymbol{x}_1})$ and $\boldsymbol{y} = \Delta_{\boldsymbol{y}} - (\delta_{\boldsymbol{y}_0} + \delta_{\boldsymbol{y}_1})$, computes $z = f_{dist}(\boldsymbol{x}, \boldsymbol{y})$ and $o = 1_{z \geqslant \theta}$.
- ***Output***: Sends o_j to P_{res}.

Theorem 2. *In the $\mathcal{F}_{\text{FUNSHADE}.setup}$-hybrid model, protocols 12-13-14 (online phase) securely realize the functionality $\mathcal{F}_{th-dist}$.*

Proof. The semi-honest adversary corrupts P_j during the sequential execution of protocols 12-13-14. For this case, $\mathcal{S}$ executes the setup phase honestly on the behalf of P_{1-j} (in case of interactive setup), and will simulate the entire circuit evaluation, assuming the circuit-inputs of P_{1-j} to be 0. In the FUNSHADE.Result protocol, $\mathcal{S}$ adjusts the shares of $\langle o \rangle$ on behalf of P_{1-j} so that $\mathcal{A}$ sees the same transcript as in the real-world protocol.

- FUNSHADE.Setup: For the offline phase, we consider it as an ideal functionality $\mathcal{F}_{\text{FUNSHADE}.setup}$, which generates the required FSS preprocessing keys and δ-shares. Since we make only black-box access to FUNSHADE.setup, its simulation follows from the security of the underlying primitive used to instantiate it (OT or HE for the ΠSS preprocessing material stemming from setupMULT of [193], generic 2PC for the FSS keys following Appendix A.2 of [38]), or alternatively a trusted party can be used.

- FUNSHADE.Share: For the instances where P_j is the owner of the values (e.g., $\mathsf{P}_j \equiv \mathsf{P}_{in_x}$), $\mathcal{S}$ has to do nothing since $\mathcal{A}$ is not receiving any messages. $\mathcal{S}$ receives $\Delta_{\boldsymbol{v}}$ from $\mathcal{A}$ on behalf of P_{1-j}. For the instances where P_{1-j} is the owner, $\mathcal{S}$ sets $v = 0$ and performs the protocol steps honestly.

- FUNSHADE.Eval: During the online phase, $\mathcal{S}$ follows the protocol steps honestly using the data obtained from the setup phase. The scalar product requires l local additions (non-interactive and thus they don't need to be simulated) and a subsequent reconstruction of $\langle \hat{z}_\theta \rangle$ as $\hat{z}_\theta = \hat{z}_{\theta 0} + \hat{z}_{\theta 1}$ that behaves just like FUNSHADE.Result and serves as input to the FSS IC gate. For the FSS IC gate, we resort to the Simulation-based security of [38] (Definition 2) to argue computational indistinguishability of the ideal and real world executions, hiding the information of r contained in $\boldsymbol{k}_0$ and $\boldsymbol{k}_1$ from $\mathcal{A}$.

- FUNSHADE.Result: To reconstruct a value $\langle o \rangle$, $\mathcal{S}$ is given the output o, which is the output of $\mathcal{A}$. Using o and the share o_{1-j} corresponding to P_{1-j}, $\mathcal{S}$ computes $o_j = o - o_{1-j}$ and sends this to $\mathcal{A}$ on behalf of P_{1-j}. $\mathcal{S}$ receives o_j from $\mathcal{A}$ on behalf of P_{1-j}.

□

Definition 4 (Correctness of Funshade)**.** *For every threshold $\theta \in \mathbb{Z}^*_{n+}$, every pair of input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^l_n$ and every function $f_{dist}(\boldsymbol{x}, \boldsymbol{y}) : \mathbb{Z}^l_n \to \mathbb{Z}_n$ from Table 5.2,*

$$\begin{aligned}
&\textit{if } (\boldsymbol{k}_0, \boldsymbol{k}_1, \langle \delta_{\boldsymbol{x}} \rangle, \langle \delta_{\boldsymbol{y}} \rangle) \leftarrow \text{FUNSHADE.Gen}(l, n, \lambda, \theta) \\
&\textit{and } (\boldsymbol{\Delta}_{\boldsymbol{x}}, \langle d_{\boldsymbol{x}} \rangle \leftarrow \text{FUNSHADE.Share}(\boldsymbol{x}, \langle \delta_{\boldsymbol{x}} \rangle), \\
&\quad \boldsymbol{\Delta}_{\boldsymbol{y}}, \langle d_{\boldsymbol{y}} \rangle \leftarrow\leftarrow \text{FUNSHADE.Share}(\boldsymbol{y}, \langle \delta_{\boldsymbol{y}} \rangle)) \\
&\textit{then } \Pr[\text{FUNSHADE.Eval}(0, \Delta_{\boldsymbol{x}}, \Delta_{\boldsymbol{y}}, d_{\boldsymbol{x}0}, d_{\boldsymbol{y}0}, \boldsymbol{k}_0) \\
&\quad + \text{FUNSHADE.Eval}(1, \Delta_{\boldsymbol{x}}, \Delta_{\boldsymbol{y}}, d_{\boldsymbol{x}1}, d_{\boldsymbol{y}1}, \boldsymbol{k}_1) \\
&\quad = 1_{f_{dist}(\boldsymbol{x},\boldsymbol{y}) \geqslant \theta}] = 1.
\end{aligned} \tag{5.3}$$

Theorem 3. *Jointly, protocol 11 (offline phase), and protocols 13-12-14 (online phase), realize the function $f(f_{dist}, \theta, \boldsymbol{x}, \boldsymbol{y}) = 1_{f_{dist}(\boldsymbol{x},\boldsymbol{y}) \geqslant \theta}$ correctly.*

Proof. We first decompose the Π-sharing based scalar product (step 1 of Protocol 13) for the joint result of the two computing parties $\hat{z}_\theta$ in Equation 5.4,

$$\begin{aligned}\hat{z}_\theta = \hat{z}_{\theta 0} + \hat{z}_{\theta 1} &= (r_{\theta 0} + r_{\theta 1}) + (d_{\boldsymbol{x}_0} + d_{\boldsymbol{x}_1}) + (d_{\boldsymbol{y}_1} + d_{\boldsymbol{y}_1}) + f_{cp} \cdot \textstyle\sum^l \\ &[\Delta_{\boldsymbol{x}}\Delta_{\boldsymbol{y}} - (\Delta_{\boldsymbol{x}}\delta_{\boldsymbol{y}_0} + \Delta_{\boldsymbol{x}}\delta_{\boldsymbol{y}_1}) - (\Delta_{\boldsymbol{y}}\delta_{\boldsymbol{x}_0} + \Delta_{\boldsymbol{y}}\delta_{\boldsymbol{x}_1}) + (\delta_{\boldsymbol{xy}_0} + \delta_{\boldsymbol{xy}_1})] \\ &= r_\theta + d_{\boldsymbol{x}} + d_{\boldsymbol{y}} + f_{cp} \cdot \textstyle\sum^l [\Delta_{\boldsymbol{x}}\Delta_{\boldsymbol{y}} - \Delta_{\boldsymbol{x}}\delta_{\boldsymbol{y}} - \Delta_{\boldsymbol{y}}\delta_{\boldsymbol{x}} + \delta_{\boldsymbol{xy}}] \\ &= r - \theta + d_{\boldsymbol{x}} + d_{\boldsymbol{y}} + f_{cp} \cdot \textstyle\sum^l [\Delta_{\boldsymbol{x}}\Delta_{\boldsymbol{y}} - \Delta_{\boldsymbol{x}}\delta_{\boldsymbol{y}} - \Delta_{\boldsymbol{y}}\delta_{\boldsymbol{x}} + \delta_{\boldsymbol{x}}\delta_{\boldsymbol{y}}] \\ &= r - \theta + d_{\boldsymbol{x}} + d_{\boldsymbol{y}} + f_{cp} \cdot \textstyle\sum^l (\Delta_{\boldsymbol{x}} - \delta_{\boldsymbol{x}}) \cdot (\Delta_{\boldsymbol{y}} - \delta_{\boldsymbol{y}}) \\ &= r - \theta + f_{local}(\boldsymbol{x}) + f_{local}(\boldsymbol{y}) + f_{cp} \cdot \textstyle\sum^l (\boldsymbol{x}^{(i)} \cdot \boldsymbol{y}^{(i)}) \\ &= r - \theta + f_{dist}(\boldsymbol{x}, \boldsymbol{y}) = z_\theta + r\end{aligned} \tag{5.4}$$

where we group all the SS shares and reconstruct their original values, replace r_θ and $\delta_{\boldsymbol{xy}}$ by the corresponding values (from definitions in protocol 1), group the Π-shares of $\boldsymbol{x}$ and $\boldsymbol{y}$ to later reconstruct their values, and finally make use of Equation 5.2.

With the public input $\hat{z}$ sorted out, we analyze the Interval Containment evaluation with output reconstruction in Equation 5.5,

$$\begin{aligned}o = o_1 + o_2 &= \text{FSS.Eval}^{IC}(0, \boldsymbol{k}_0^{IC}, \hat{z}_\theta) + \text{FSS.Eval}^{IC}(1, \boldsymbol{k}_1^{IC}, \hat{z}_\theta) \\ &= \text{FSS.Eval}^{IC}(0, \text{FSS.Gen}^{IC}(\lambda, n, r)^{(0)}, z_\theta + r) \\ &\quad + \text{FSS.Eval}^{IC}(1, \text{FSS.Gen}^{IC}(\lambda, n, r)^{(1)}, z_\theta + r) \\ &= 1_{z_\theta \in \mathbb{Z}^*_{n+}} = 1_{0 \leqslant z - \theta} = 1_{f_{dist}(\boldsymbol{x}, \boldsymbol{y}) \geqslant \theta}\end{aligned} \tag{5.5}$$

where we resort to Theorem 3 of [38] to argue that the two protocols (FSS.Gen$^{IC}(\lambda, n, r)$, FSS.Eval$^{IC}(j, \boldsymbol{k}_j^{IC}, \hat{z}_\theta)$) constitute an FSS gate[5]correctly realizing $f(z_\theta) = 1_{p \leq z_\theta \leq q}$. Then, following Definition 2 (Correctness) of [38], we can argue that Pr[FSS.Eval$^{IC}(0, \boldsymbol{k}_0^{IC}, \hat{z}_\theta)$ + FSS.Eval$^{IC}(1, \boldsymbol{k}_1^{IC}, \hat{z}_\theta) = 1_{0 \leq z_\theta \leq 2^{n-1}-1}] = 1$, thus equating the output of the FSS gate to $1_{z_\theta \in \mathbb{Z}^*_{n+}}$, the unit step function. □

5.3.5 Experiments

We implement our solution in a standalone Python library with efficient C++ blocks by virtue of Cython. Our code is available at `https://github.com/ibarrond/funshade`. We use a Miyaguchi-Preneel one-way compression function with an AES block cipher for our PRG construction, an extended variant of Matyas-Meyer-Oseas function used in previous works [213]. We concatenate several fixed key block ciphers to achieve the desired output length.

We timed the execution of FUNSHADE.Eval in a single computing party to 900μs with one single core (Processor AMD Ryzen 5 PRO 3500U, 2100 Mhz, 4 Cores available), and around 550μs when using two cores to speed up the Interval Containment evaluation (one DCF per core). This indicates that the communication latency (e.g., 10ms for LAN, 70ms for WAN)

[5]There are several notation elements to adapt in order to align with [38]. Our mask r is written as r^{in} in Figure 3 of [38] depicting the FSS IC gate. We set the parameters $p = 0$ and $q = 2^{n-1} - 1$ to define the interval containing all positive integers. $1_{p \leq z_\theta \leq q} = g_{IC,n,p,q}(z_\theta)$ is a function that belongs (per definition of IC gate in Section 4 of [38]) to the family of functions $\mathcal{G}^{IC}_{n,p,q}$ referenced in Theorem 3 of [38].

Table 5.3: Benchmark of theoretical costs on evaluating a scalar product and comparison to threshold between two vectors of size l with n-bit integers

Work	Type	#Rounds of communication	#ring elements in communication	Correctness	Online Computation Blocks
AriaNN [213]	2PC SS: Arith., FSS	2 (1+1)	$4l + 4$	N	SS scalar product, FSS Comparison (1 DCF)
Boyle et. al. [38]	2PC SS: Arith., FSS	2 (1+1)	$4l + 4$	Y	SS scalar product, FSS IC gate (2 DCF)
ABY [88]	2PC SS: Boolean&Arith, GC	3 (1+2+0)	$\gg 6l$	Y	SS scalar product, Arith. to Yao conversion, GC evaluation
ABY2.0 [193]	2PC ΠSS: Boolean&Arith.	5 (1+1+3)	$\gg 2$	Y	ΠSS scalar product, Arith. to Boolean conversion, BitExtraction
GSHADE [44] (only scalar prod.)	2PC OT	2	$> 2l$	Y	correlated OTs.
CryptFlow2 [202]	2PC SS: Arith., OT	5	$> (128 + 14)l$	Y	Linear layer (1-dim weights), dReLU
Falcon [240]	3PC Replicated SS: Arith.	8 (1+7)	> 6	Y	MatMult with 1-dim matrices, Private Compare
FUNSHADE (OURS)	**2PC** **ΠSS: Arith., FSS**	**1**	**2**	**Y**	**ΠSS scalar product,** **FSS IC gate (2 DCF)**

would be the main bottleneck in a real-world deployment for 1:1 distance calculations, and there would be a wide margin to compensate communication with computation in 1:N or M:N scenarios (e.g., biometric identification). We also vary the vector sizes ranging from $l = 64$ to $l = 65536$, with negligible impact to the FUNSHADE.Eval time, indicating that the main bottleneck in terms of computation is located in the evaluation of DCFs.

Additionally, we test our solution with randomized input vectors for all distance metrics, verifying the 100% correctness as long as natural overflows ($z > 2^{n-1} - 1$ or $z < -2^{n-1}$) are avoided.

5.3.6 Related Work

Distance metric evaluations, specially for Hamming Distance and Scalar Products, range among the most typical applications of privacy-preserving computation techniques. Consequently, a wide variety of previous work in MPC, FHE and FE have dealt with some form of it.

The Multi Party Computation field includes a plethora of works covering distance metric evaluations. All the frameworks for privacy preserving neural networks cover scalar-product-based matrix multiplications often followed by ReLU activations [158, 29, 208, 78, 240], covering a mixture of Garbled Circuits, Secret Sharing and their conversions. Secure hamming distance evaluation has motivated work such as [45] based on Oblivious Transfer, with its generalization to multiple metrics in [44]. Mixed-mode protocols have also tackled distance evaluations [88, 185, 193]. However, the majority of these solutions incur in a considerable communication cost to perform comparison. More recently, solutions based on FSS [213, 38, 41] have shown promising results, leading to this work.

We compare the online phase performance of our solution with that of selected previous works in Table 5.2. FUNSHADE is the first work in the 2PC setting requiring one single round of communication to evaluate $1_{x^T y > \theta}$ while also presenting the lowest communication size of 2 ring elements. An additional side-by-side comparison with AriaNN [213] is provided in Appendix A.3.

5.3.7 Conclusions

In this section we presented FUNSHADE, a novel 2PC privacy-preserving solution of various distance metrics (e.g., Hamming distance, Scalar Product) followed by threshold comparison. We built this protocol upon an optimized version of arithmetic secret sharing for the secure evaluation of linear operations and functional secret sharing with 100% correctness for comparison. Thanks to this, FUNSHADE proposes the first solution in the 2PC literature requiring one single round of communication in the online phase while outperforming all previous works in online communication size (two ring elements), all while relying only in lightweight cryptographic primitives. We implement our solution from scratch in Python with efficient C++ blocks, and test it to record a runtime of less than 1ms per computing party excluding communication costs.

5.4 GROTE: Group Testing for Privacy-Preserving Biometric Identification [131]

Abstract This section describes GROTE, a novel method to perform privacy-preserving face identification based on the notion of group testing, and applies it to a solution using the Cheon-Kim-Kim-Song (CKKS) homomorphic encryption scheme. Securely computing the closest reference template to a given live template requires K comparisons, as many as there are identities in a biometric database. GROTE replaces element-wise testing by group testing to drastically reduce the number of such costly, non-linear operations in the encrypted domain from K to up to $2\sqrt{K}$. More specifically, we approximate the max of the coordinates of a large vector by raising to the α-th power and cumulative sum in a 2D layout, incurring a small impact in the accuracy of the system while greatly speeding up its execution. We implement GROTE and evaluate its performance.

5.4.1 Introduction

In client-server scenarios like biometric identification or ML-as-a-service (MLaaS), FHE shines at offloading heavy computation almost exclusively to one party[6]. Moreover, by employing well established schemes like Brakerski-Gentry-Vaikuntanathan (BGV [43]), Brakerski / Fan-Vercauteren (BFV [100], described in Section 3.3.3) or Cheon-Kim-Kim-Song (CKKS [62], covered in Section 3.3.4), we obtain private computation capabilities suited for biometric operations (R1), protecting the biometric templates (R3) and their matchings (R2). Standard FHE provides privacy-preserving guarantees following an *Honest - But -*

[6]The encryption and decryption of data still needs to be performed by the data owner.

Curious threat model where parties involved perform the selected protocol without deviation while attempting to obtain as much information from the private data as possible. However, FHE only offers out-of-the-shelf encrypted addition and multiplication. Non-linear operations such as comparisons must be either reformulated using ring properties as in BFV & BGV [138] or approximated with polynomials in a small interval [64, 63, 164], incurring in a loss of numerical precision (CH1). In both cases, several ciphertext-to-ciphertext multiplications are required to compute an encrypted comparison, considerably increasing its computational cost and that of comparison-based operations (e.g., Rectified Linear Units (ReLU), maximum of an array). Since biometric identification systems require multiple comparisons (one per record held in the biometric database of reference), reducing the cost of this operation directly improves the practicality of FHE to protect these systems in real-world deployments (R6).

Our Contributions. We propose a novel method to perform privacy-preserving biometric identification based on the notion of group testing, and instantiate it on FHE with the CKKS scheme. Securely computing the closest reference template to a given live template requires K comparisons, as many as there are identities in a biometric database. Our solution, named GROTE, replaces element-wise testing by group testing to reduce the number of such costly, non-linear operations in the encrypted domain. More specifically, we approximate the max of the coordinates of a large vector (its infinity norm) by raising to the α-th power and cumulative sum (its α norm) in a 2D layout, incurring a small impact in the accuracy of the system while greatly speeding up its execution (1.5 times faster). We implement CKKS-based GROTE and show that it outperforms the straightforward alternative based on batched comparisons.

GROTE is arranged as follows. Section 5.4.2 details our design of an FHE-enabled privacy-preserving group testing solution. Section 5.4.3 instantiates it to the CKKS scheme. Next we validate this approach with experiments on biometric data in Section 5.4.4. We conclude the section with a review of previous works in Section 5.4.5 and some takeaways in Section 5.4.6.

5.4.2 An Idea: use Group Testing for the Threshold Comparison

The notion of group testing emanates from the field of statistics [93], and has been applied extensively across industries (e.g., in the healthcare sector [116], or in fault detection [173]). The essence of group testing consists of performing a check in a group of samples all at once, rather than checking on individual samples. For example, in the biometric identification domain one must compare each of the similarity scores resulting from $1 : K$ matchings to a defined threshold θ, or alternatively compute the max of the vector of scores and test if this element is above the threshold. Figure 5.7 illustrates our proposal of a biometric matching algorithm based on group testing, that we name GROTE.

The main insight that drives our solution is that, as proposed in [110], for a sufficiently large exponent α we can approximate the max operation of Eq. 2.3 (also expressed as the infinity norm $\|bmz\|_\infty$) by the α norm:

Algorithm 15 GROTE($\mathbf{z}$, h, w, θ_w,θ_h, α) $\rightarrow k^*$

```
Players: z, a vector of size $K$ holding the similarity scores of a live template with each of
    the $K$ reference templates. ,
        $h$, number of rows (or size of columns) of the group testing 2D matrix.
        $w$, number of columns (or size of rows) of the group testing 2D matrix.
        $\theta_w$, a threshold for row-wise comparison.
        $\theta_h$, a threshold for column-wise comparison.
        $\alpha$, exponent for max approximation.
Output: Index $k^*$ of the single score above the thresholds in the flattened score vector, set
    to zero if zero or several scores above the thresholds.
Assumption: With overwhelming probability there are either zero or one elements in z
    above $\theta$.
1: $\mathbf{z} \in \mathbb{R}^K \rightarrow \mathbf{Z} \in \mathbb{R}^{h \times w}$. Reshape vector $\mathbf{z} \in \mathbb{R}^K$ as matrix $\mathbf{Z} \in \mathbb{R}^{h \times w}$. Fill empty spaces
    with zeros.
2: $\mathbf{Z} \rightarrow \mathbf{Z}^\alpha$. Raise each element of $\mathbf{Z}$ to the $\alpha$ power.
Cumulative Sum:
3: $\overrightarrow{\mathbf{w}} : \mathbf{w}[i] = \sum_{j=1}^{w} \mathbf{Z}^\alpha[i,j]$. Compute $\overrightarrow{\mathbf{w}} \in \mathbb{R}^h$, the row-wise sum of $\mathbf{Z}^\alpha$.
4: $\overrightarrow{\mathbf{h}} : \mathbf{h}[j] = \sum_{i=1}^{h} \mathbf{Z}^\alpha[i,j]$. Compute $\overrightarrow{\mathbf{h}} \in \mathbb{R}^w$, the column-wise sum of $\mathbf{Z}^\alpha$.
Comparison:
5: $\overrightarrow{\mathbf{w}}_{\theta_w} : \overrightarrow{\mathbf{w}}_{\theta_w}[i] = (\overrightarrow{\mathbf{w}}[i] \geq \theta_w)_?$. Compare elements of $\overrightarrow{\mathbf{w}}$ to threshold $\theta_w$.
6: $\overrightarrow{\mathbf{h}}_{\theta_h} : \overrightarrow{\mathbf{h}}_{\theta_h}[j] = (\overrightarrow{\mathbf{h}}[j] \geq \theta_h)_?$. Compare elements of $\overrightarrow{\mathbf{h}}$ to threshold $\theta_h$.
Validation.
7: Compute sums $v_w = \Sigma \overrightarrow{\mathbf{w}}_{\theta_w}$ and $v_h = \Sigma \overrightarrow{\mathbf{h}}_{\theta_h}$.
8: Compute $v_K = (v_h \cdot v_w \leq 1)_?$, check if up to one non-zero element in 2D matrix layout.
ArgMax.
9: $i^* = \Sigma_{i=1}^{w}(i \cdot \mathbf{w}_{\theta_w}[i])$. Compute $i^*$, row index of the above-threshold score.
10: $j^* = \Sigma_{j=1}^{h}(j \cdot \mathbf{h}_{\theta_h}[j])$. Compute $j^*$, column index of the above-threshold score.
11: $k^* = (wi^* + j^*)$. Compute the index of the above-threshold score.
12: return either $k^*$ and $v_K$ separately or $k^* \cdot v_K$.
```

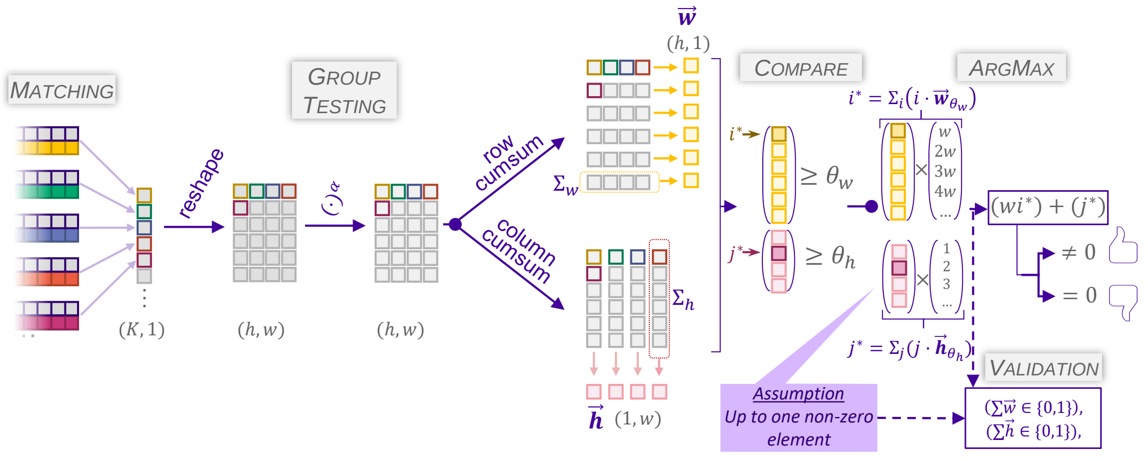


Figure 5.7: Group testing diagram

$$\max(bmz) = \|bmz\|_\infty \approx \|bmz\|_\alpha = \sqrt[\alpha]{\Sigma_{i=1}^{K}(bmz[i]^\alpha)} \tag{5.6}$$

Moreover, we can turn it into a linear operation by removing the root and tweaking the threshold θ:

$$(\|bmz\|_\infty \geq \theta)_? \approx (\Sigma_{i=1}^{K}(bmz[i]^\alpha) \geq \theta^\alpha)_? \tag{5.7}$$

This approximation is more precise the higher the exponent α, and is rendered less precise the more elements there are in the vector. To balance out, we resort to pooling (aggregating) parts of the bmz vector inspired by *group testing*: we reshape $bmz \in \mathbb{R}^K$ into a 2D matrix[7] $bmZ \in \mathbb{R}^{h\times w}$, obtaining h rows and w columns, and use Eq. 5.8 to approximate the maximum value row-wise and column-wise:

$$\begin{aligned} b\vec{m}w &\triangleq b\vec{m}w[i] = \sum_{j=1}^{w}(bmZ[i,j])^\alpha \ \ \forall i \\ b\vec{m}h &\triangleq b\vec{m}h[j] = \sum_{i=1}^{h}(bmZ[i,j])^\alpha \ \ \forall j \end{aligned} \tag{5.8}$$

We then resort to standard comparisons and ArgMax to pin-point the element yielding a positive score, if any. The threshold θ, tied to the binary classification task arising from the biometrics scenario, must be tweaked to account for the pooling operation and the exponent α. For that purpose, we define two new group-wise thresholds θ_w and θ_h that must be set with properly adapted biometric-based experiments (see Section 5.4.4 for an example of such setting), and we use them to compare the row-wise and column-wise groupings $b\vec{m}w \geq \theta_w$ and $b\vec{m}h \geq \theta_h$ respectively:

[7]Pooling with higher dimensionality is possible and straightforward to derive from our construction. We employ 2D pooling in our solution to allow for descriptive visuals.

$$\begin{aligned}
b\vec{mw}_{\theta_w} &\triangleq b\vec{mw}_{\theta_w}[i] = (b\vec{mw}[i] \geq \theta_w)_? \;\; \forall i \\
b\vec{mh}_{\theta_h} &\triangleq b\vec{mh}_{\theta_h}[j] = (b\vec{mh}[j] \geq \theta_h)_? \;\; \forall j \\
i^* &\triangleq \arg\max_i \;\; b\vec{mw}_{\theta_w}[i] \\
j^* &\triangleq \arg\max_j \;\; b\vec{mh}_{\theta_h}[j]
\end{aligned} \tag{5.9}$$

If the two comparisons yield a positive result at some indices i^* and j^* respectively, we conclude that the element $bmZ[i^*, j^*]$ and the element $bmz[w(i^*) + (j^*)]$ in the unrolled score vector contain a positive score, yielding the identity $w(i^*) + (j^*)$ as result. Otherwise, we conclude that no positive score was found, and return a negative result.

Since $h \cdot w \approx K$, we achieve a reduction in the number of comparisons from K (max of K values) to $h + w$ (max on each dimension). In the most balanced case where $h \approx w$, we need $2\sqrt{K}$ comparisons, $\sqrt{K}/2$ times less comparisons than in the naïve solution.

The second insight we consider is that in the biometrics domain there is a very low chance of two or more simultaneous hits in a biometric database. This arises from the fact that the feature extractors are designed to separate (with respect to the similarity metric) templates belonging to different identities, and the reference database can be designed to minimize the likelihood of this event [8].

Relying on this fact, we can obtain the index of the non-zero value (arg max) that represents the positive identity k^* (if any) by multiplying the results of the elementwise comparison $b\vec{mw}_\theta$ and $b\vec{mh}_\theta$ with non-overlapping indexing vectors and summing up all the elements:

$$\begin{aligned}
&(HW(b\vec{mw}_{\theta_w}) \leq 1) \;\wedge\; (HW(b\vec{mh}_{\theta_h}) \leq 1) \Rightarrow \\
&k^* = w \cdot i^* + j^* = \sum_{i=1}^{h} w \cdot i \cdot b\vec{mw}_{\theta_w}[i] + \sum_{j=1}^{w} j \cdot b\vec{mh}_{\theta_h}[j]
\end{aligned} \tag{5.10}$$

As an additional precaution, we add an optional validation check to ensure that there is indeed up to a single non-zero element in each of the results of the elementwise comparison $b\vec{mw}_\theta$ and $b\vec{mh}_\theta$. Since a positive hit requires to have a non-zero element in both $b\vec{mh}_\theta$ and $b\vec{mw}_\theta$, we can reduce the check to:

$$\begin{aligned}
&\text{valid}(b\vec{mw}_{\theta_w}, b\vec{mh}_{\theta_h}) = \\
&\qquad (HW(b\vec{mw}_{\theta_w}) \leq 1) \;\wedge\; (HW(b\vec{mh}_{\theta_h}) \leq 1) = \\
&\qquad \left(\sum b\vec{mw}_{\theta_w}[i]\right) \cdot \left(\sum b\vec{mh}_{\theta_h}[j]\right) \leq 1
\end{aligned} \tag{5.11}$$

We detail the step-by-step group testing algorithm in Algorithm 15.

[8] One could measure the distance among reference templates while building the database, and reject/-modify new reference templates that are too close to existing ones.

5.4.3 Applying GROTE to CKKS

In this section, we detail the steps to instantiate our group testing algorithm to CKKS. To this end, there are several aspects to take into consideration:

- All encrypted operations (additions, multiplications, comparisons) happen in SIMD fashion, applied simultaneously to all the elements of the underlying encoded vector. Given that CKKS ciphertexts encoded in a ring of polynomial degree n can hold $n/2$ floating-point values, there is room to encode multiple reference templates of length l per ciphertext ($n/2l$ templates per ciphertext to be precise, where $l < n/2$). To perform multiple scalar products at once during the matching phase, we encrypt a $n/2l$ times repetition of the live template inside the input ciphertext and operate on all repetitions at once. Similarly, the comparison operation will be applied elementwise to up to $n/2$ matching scores.
- A cumulative addition of s slots inside a ciphertext can be performed by iteratively rotating and adding a ciphertext with himself $\log_2(s)$ times.
- Ciphertext to ciphertext $c \times c$ encrypted multiplications are the costliest linear operations in CKKS both in terms of noise growth and in computational time. Thus, optimizing the GROTE algorithm's runtime in CKKS involves minimizing the number of such operations. Some trade-offs to consider are:
  - Use of either non-encrypted live templates (sacrificing live template privacy) or non-encrypted reference templates (sacrificing the privacy of the reference template database), employing cheaper ciphertext-plaintext $c \times p$ multiplications for the scalar products of the matching step, also saving up in relinearization.
  - Keeping a low value of the exponent α, thus incurring in $\log_2(\alpha)$ multiplications, at the expense of a less precise approximation of the max operation.
  - Using a low number of multiplications (`depth` in [164]) in the polynomial approximation of the $sgn(x)$ function, in exchange for noisier results.
- Operating between ciphertexts requires them to have the same scale. To rescale ciphertexts (by mod-switching or multiplying with a plaintext) we will rely on the low-error techniques from [150].

5.4.3.1 Threat Model and Security Analysis

We consider a threat model whereby both the users and the Identification Server behave Semi-honestly, that is, they perform the computations faithfully while trying to obtain as much information as possible. We assume no collusion between the users, the Identification Server (IS) and the Biometric Provider (BP) (see Figure 5.8 or Section 5.4.3.2 for a description of the IS and BP).

The Biometric Provider must be trusted in the enrollment phase, as he is in charge of building the reference DB. Therefore, we can rely on the BP to perform the CKKS key generation, encrypt the DB and hold the CKKS secret key, decrypting the results sent by the Identification Server.

While the querying users seek to preserve the privacy of their live templates from the IS and the BP, the BP seeks to preserve the privacy of the reference DB from the IS and the querying users. Privacy of the inputs and intermediate computation results is assured by the use of CKKS thanks to the hardness of the LWE problem.

5.4.3.2 The end-to-end identification protocol

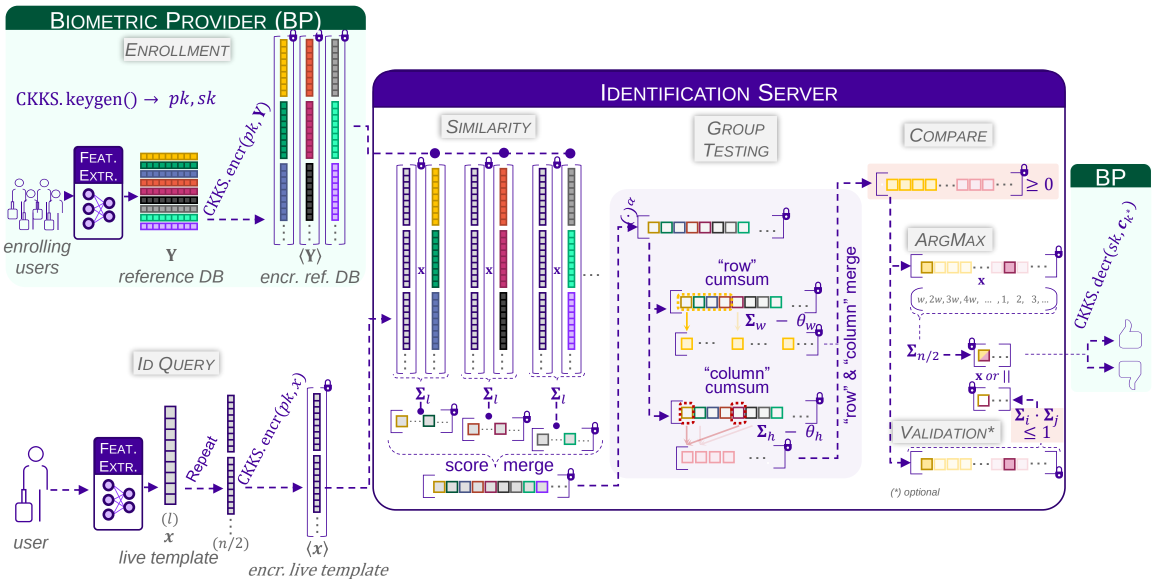


Figure 5.8: CKKS group testing

Building upon the Grote algorithm, we design a protocol for privacy-preserving biometric identification based on CKKS, depicted in Figure 5.8. As *setup* for our scenario, the Biometric Provider (BP) acts as trusted entity and collects the reference templates, generating a pair of public and secret keys, and encrypting the reference template database with SIMD for compression (with $n/2l$ ref. templates per ciphertext). This encrypted database is deployed to an Identification Server (IS), in charge of the full encrypted computation, while the public key is then distributed to the users.

A user wishing to identify himself extracts his live template, encrypt it with $n/2l$ repetitions into a single ciphertext bmc_x, and then queries the IS with it. The server performs the following steps:

1. *Similarity*: Compute the scalar product between the live template ciphertext $\langle bmx \rangle$ and every ciphertext in the encrypted DB of reference templates $\langle \{bmY[1], \ldots, bmY[n/2l]\} \rangle$, $\ldots$, $\langle \{\ldots, bmY[K]\} \rangle$. Making use of SIMD multiplications followed by cumulative additions ($\log_2(l)$ rotations and additions per DB ciphertext), the server obtains K similarity scores (one per record) distributed evenly among $\lceil 2Kl/n \rceil$ ciphertexts $\langle bmz' \rangle_1$, $\ldots$, $\langle bmz' \rangle_{\lceil 2Kl/n \rceil}$.

2. *Score merge*: merge of the score ciphertexts by multiplying with masking plaintexts (vectors with ones in the slots containing scores, zeros elsewhere), and then adding all the masked scores into $T = \lceil 2K/n \rceil$ ciphertexts $\langle bmz \rangle_1, \ldots, \langle bmz \rangle_T$.

3. *Group testing*: as described in Algorithm 15, to approximate the *max* by a sum of α-powered values in a 2D matrix layout. This involves, per each of the T score ciphertexts, $\log_2(\alpha)$ $c \times c$ multiplications, two cumulative additions for the "row-wise" and "column-wise" vectors (using $\log_2(h)$ and $\log_2(w)$ rotations & additions respectively), the subtraction of their respective thresholds θ_w and θ_h and their merging (as in step 2) into a single ciphertext.

4. *Comparison*: with zero carried out following the procedure described in [163]. This involves $\log_2(\texttt{depth})$ multiplications and additions.

5. *Argmax*: by multiplying with constant index vectors and a cumulative sum to obtain the identity (if any) of the live template's provider.

6. *Validation*: Since comparisons are too expensive to justify one for validation, we are left with two alternatives:

    - Dropping the Argmax step entirely and decrypting the comparison result directly.
    - Performing the cumulative sum of all the elements resulting from the comparison in each vector ($\log_2(max(h, w))$ rotations and additions), multiplying the two results together and outputting it alongside the Argmax result.

A naïve solution would require K similarity computations and K comparisons (plus the Argmax and validation steps), whereas adding the *group testing* step reduces the number of comparisons to $h + w$ in exchange for $log_2(\alpha)$ multiplications and a cumulative addition. As we discuss in Section 5.4.3.3, the approximated comparison from [164] requires far more multiplications (≥ 11) than *group testing* (for $\alpha \leq 32$). Crucially, due to the SIMD feature of CKKS, the GROTE save-up kicks in for $K > n/2$, since otherwise a single comparison would suffice for the identification and the group testing step would be redundant.

We detail the full CKKS-based computation in Algorithm 16.

5.4.3.3 Choosing Parameters

The GROTE related parameters h and w will be set based on performance experiments in Section 5.4.4, whereas α will be tested for values $\alpha \in \{2, 4, 8, 16, 32\}$.

The biometric template size l is set by the architecture of biometric feature extractors. In the face biometrics domain, they often amount to $l \in \{128, 256, 512\}$ to speed up non-encrypted similarity score calculations (e.g. using AVX instructions for the multiplication). We will use the smaller $l = 128$ to maximize the number of reference templates per ciphertext, and thus the number of comparisons per ciphertext.

The CKKS scheme parameters n (polynomial ring degree) and q (modulus of the polynomial coefficients) are tied to each other and linked to the sought-out security parameter. To obtain an equivalent security of 128 bits, and according to [16], $n = 16384$ allows

Protocol 16 CKKS.GROTE($\langle \mathbf{x} \rangle$,$\{\langle \mathbf{Y} \rangle\}$, h, w, θ_w,θ_h, α)$\rightarrow \langle \mathbf{k}^* \rangle$

Players: $\{\langle \mathbf{Y} \rangle\} = \{\langle \mathbf{Y}[1 \dots n/2l] \rangle, \dots, \langle \mathbf{Y}[\dots K] \rangle, \}$, an encrypted database of K reference templates of length l split among $\lceil 2Kl/n \rceil$ ciphertexts.
$\langle \mathbf{x} \rangle$, the encrypted live template of length l repeated $n/2l$ times.
h, number of rows (or size of columns) of the group testing 2D matrix.
w, number of columns (or size of rows) of the group testing 2D matrix.
θ_w, a threshold for row-wise comparison.
θ_h, a threshold for column-wise comparison.
α, exponent for max approximation.

Output: Encrypted Index $\langle k^* \rangle$ of the single score above the thresholds in the flattened score vector, set to zero if no match in both dimensions

Assumption: With overwhelming probability there are either zero or one elements in $\mathbf{z}$ above θ.

```
*Similarity*:
1: for a in 1...$\lceil 2Kl/n \rceil$ do
2:     $\langle \mathbf{z}' \rangle^{(a)} = \langle \mathbf{Y} \rangle^{(a)} \cdot \langle \mathbf{x} \rangle$
3:     for i in 1...$l$ do
4:         $\langle \mathbf{z}' \rangle^{(a)} += (\langle \mathbf{z}' \rangle^{(a)} \ll i)$
*Score Merge*:
5: for t in 1...$\lceil 2K/n \rceil (= T)$ do
6:     for i in 1...$l$ do
7:         $\langle \mathbf{z} \rangle^{(t)} = \langle \mathbf{z}' \rangle^{(t \cdot 2Kl/n+i)} \cdot \mathbf{mask}_{n/2}(2l/n+i)$
*Group Testing*:
8: for t in 1...$T$ do
9:     for _ in 1...$\log_2 \alpha$ do
10:        $\langle \mathbf{z} \rangle^{(t)} = \langle \mathbf{z} \rangle^{(t)} \cdot \langle \mathbf{z} \rangle^{(t)}$
    $\langle \overrightarrow{\mathbf{w}} \rangle = -\theta_w$; $\langle \overrightarrow{\mathbf{h}} \rangle = -\theta_h$
11: for t in 1...$T$ do
12:     $\langle \overrightarrow{\mathbf{w}} \rangle += \sum_{i=1}^{n/2w}((\langle \mathbf{z} \rangle^{(t)} \cdot \mathbf{mask}_{n/2}(wi)) \ll wi)$.
13:     $\langle \overrightarrow{\mathbf{h}} \rangle += \sum_{h=1}^{n/2h}((\langle \mathbf{z} \rangle^{(t)} \cdot \mathbf{mask}_{n/2}(j)) \ll j)$.
14: $\langle \mathbf{z}_{Grote} \rangle = concat(\langle \overrightarrow{\mathbf{w}} \rangle, \langle \overrightarrow{\mathbf{h}} \rangle)$
*Comparison*:
15: $\langle \mathbf{z}_\theta \rangle = \mathrm{OptMinimaxComp}(\langle \mathbf{z}_{Grote} \rangle, 0, [\dots])$
*ArgMax.*
16: $\langle k^* \rangle = \sum_{h+w}(\langle \mathbf{z}_\theta \rangle \cdot \{w, 2w, \dots, hw, 1, 2, \dots, w-1\}$.
*Validation (optional)*:
17: $\langle v \rangle = \sum_w \langle \mathbf{z}_\theta \rangle \cdot ((\sum_h \langle \mathbf{z}_\theta \rangle) \ll w)$,
18: return either $(\langle k^* \rangle, \langle v \rangle)$ or $\langle \mathbf{z}_\theta \rangle$.
```

$\log_2 q \leq 438$ bits and $n = 32768$ allows $\log_2 q \leq 881$ bits[9]. Setting q is directly related to the number of multiplications (depth) of the full arithmetic circuit d, and standard strategies to set it [27, 167, 164, 166] consist of composing a chain of primes $\{q_i\}$ $\forall i \in \{1, \dots, d\}$ such that $q = \prod_{i=1}^{d+1} q_i$, with $q_1 = q_{d+1} \approx 2^{60}$ to ensure high precision in encoding/decoding and

$q_2, \ldots, q_d$ chosen to be close to the CKKS encoding scale Δ to reduce rounding errors when performing rescaling/mod-switching. Smaller values of Δ yield less precise approximations of the $sgn(x)$ function [164], but also reduce required total size of q to the point where it might permit the use of lower n (reducing the ciphertext sizes and thus speeding up their operations). As such, we find a good trade-off in setting $\Delta = 2^{30}$, which drives us to set $q \approx 2^{120} \cdot 2^{30d}$ (permitting $d \leq 10$ for $n = 16384$ and $d \leq 25$ for $n = 32768$).

To define d, we need to count the total amount of multiplications. We require one multiplication for the *similarity* computation and one for the merging, $\log_2(\alpha)$ multiplications for the *group testing* step plus one for the extra merging. Based on the accuracy of the approximation of the $sgn(x)$ function in Table V of [164] we employ `depth` $= 11$ to get a wide margin $\eta = 2^{-26}$, thus requiring 11 multiplications per comparison. One last multiplication is required for the argmax operation (and optionally one more for the validation step). We thus set $d = 1 + 1 + \log_2(\alpha) + 1 + 11 + 1 = 15 + \log_2(\alpha)$, leading us to confidently set $n = 32768$ for $\alpha \leq 32$.

5.4.3.4 On (not) applying Grote to BFV or TFHE

While other FHE schemes could, on the surface, benefit from the Grote approach, we argue that the CKKS scheme is the most suitable for this application.

The BGV/BFV schemes [43, 100] operate on integers, thus computing a value z^α requires a lot of space in the ciphertext to avoid the modulo kicking in, forcing costly parameter selection in detriment of speed. Besides, the state of the art encrypted comparison techniques [138] are not suited for full SIMD computation as they require multi-slot encoding, thus rendering the scalar product more expensive and complex than in CKKS.

The TFHE scheme [65] deals with bit-level homomorphic operations, thus needing a lot of ciphertexts to encode elements with high precision from the biometric templates, a requirement to maintain high accuracy in the system. This makes big integer operations very costly, thus rendering a z^α raising operation prohibitively expensive. Besides, the TFHE scheme is not optimized for SIMD computation, thus a TFHE-based biometric identification solution would require a sizeable horizontal scaling in the hardware to achieve the same performance as CKKS.

5.4.4 Experiments

We implement our solution using the Pyfhel [136] Python library, with the SEAL [219] C++ library acting as backend. We use an ArcFace based [89] feature extractor[10]with templates of size $l = 128$. The experiments were run in an Intel(R) Core(TM) i7-7800X CPU and averaged over at least 10 runs.

We measure the biometric precision with face identification benchmarks using the Labeled Faces in the Wild (LFW) dataset [127] consisting of 13233 112x112px real face images of famous people. We employ the widespread False Acceptance Rate (FAR) and False

[9]For a secret key ternary distribution, with coefficients sampled from $\{-1, 0, 1\}$. Other distributions offer similar limitations.

[10]ArcFace-based [89] feature extractors with comparable latency and precision can be obtained from `https://github.com/deepinsight/insightface/wiki/Model-Zoo`

Rejection Rate (FRR) as metrics [141]. Typically, robust identification systems enforce $FAR \leq 10^{-3}$, obtaining a corresponding higher FRR.

Following the same procedure as for the threshold θ in standard biometric solutions, we set the thresholds θ_w and θ_h by calibrating a binary classifier with the outputs of the aggregated groups/pools used to identify random samplings of the LFW dataset, as well as their corresponding ground truth values. We train it with 8M negative samples (a live template with no hit in the DB) and 500k positive samples (a live template with a single hit in the DB). To benefit from convenient data alignment, and given the fact that both $n/$ (the number of slots) and l (the number of elements per template) are powers of 2, we test pools of the form $w, h \in 2^{\beta} \ \ \forall \beta \in \{1, \ldots, 11\}$. The biometric precision for a given pool size is applicable both vertically and horizontally. To estimate the combined error rate it suffices to combine the errors for selected w and h.

Parameter selection. Following the analysis from 5.4.3.3, we pick $n = 2^{15}$ to allow for a high enough number of multiplications. We set the modulus chain $q \approx 2^{60} * 2^{30*d} * 2^{60}$ with the maximum number of multiplications d required for the entire face identification algorithm, yielding smaller q and thus faster operations for lower circuit depth. We use templates with $l = 128$ coming out of the unmodified feature extractor.

To select K we highlight that, in order to make the Grote-based face identification solution more performant than the naïve solution, we need a reference database size $K \geq n/2 + 1$ so that a naïve face identification algorithm requires at least $T = \lceil 2K/n \rceil \geq 2$ ciphertexts to hold all the score results. By employing a synthetic augmentation of the LFW dataset [11]we are able to set $T = 2$.

5.4.4.1 Results

We first analyze the impact of group testing in the biometric **precision** of the system by first analyzing the 1D layout case (setting $w = 1$ and playing with h or vice-versa). We run face identification experiments employing 15 and record the FRR (probability of a registered user to not match with the database) for a fixed $FAR = 10^{-3}$ (probability of a non-registered user to match with the DB). As seen in Figure 5.9, and in line with our expectations, higher α yields a better max approximation, and with it lower errors. We observe that $\alpha \geq 16$ yields $FRR < 5\%$ for group sizes of up to 512, a small impact in the error that allows us to conclude that the Grote approach has a small impact in the system performance in that range.

We extend the biometric precision analysis to the full 2D layout with the same approach in Figure 5.10. We can spot balanced configurations ($w \approx h$) that preserve the biometric accuracy of the system with $FRR < 5\%$, and that allow us to confirm that the Grote preserves the biometric accuracy of the system.

To assess the **latency** gains of Grote, we also record the time required to compute each operation in the encrypted domain, as well as the runtime of the entire system, comparing the runtime of the naïve solution with that of a Grote-based solution. The results are

[11]We generate 3 randomly perturbed templates per identity that are statistically close to the original reference templates of such identity, and ensure they follow the same distribution of matching probabilities from the original LFW. This bumps the number of identities from 5749 to $K = 22996$, yielding $T = 2$.

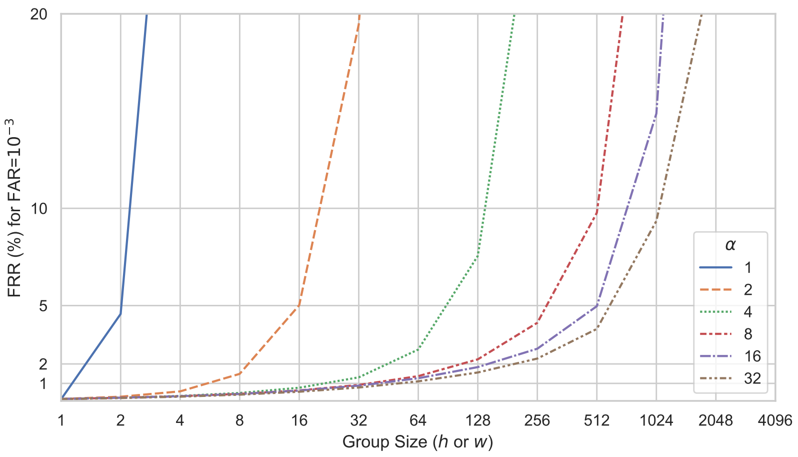


Figure 5.9: Face identification precision (False Rejection Rate at a fixed False Acceptance Rate) of 1D group testing based on the group/pool size and the exponent α.

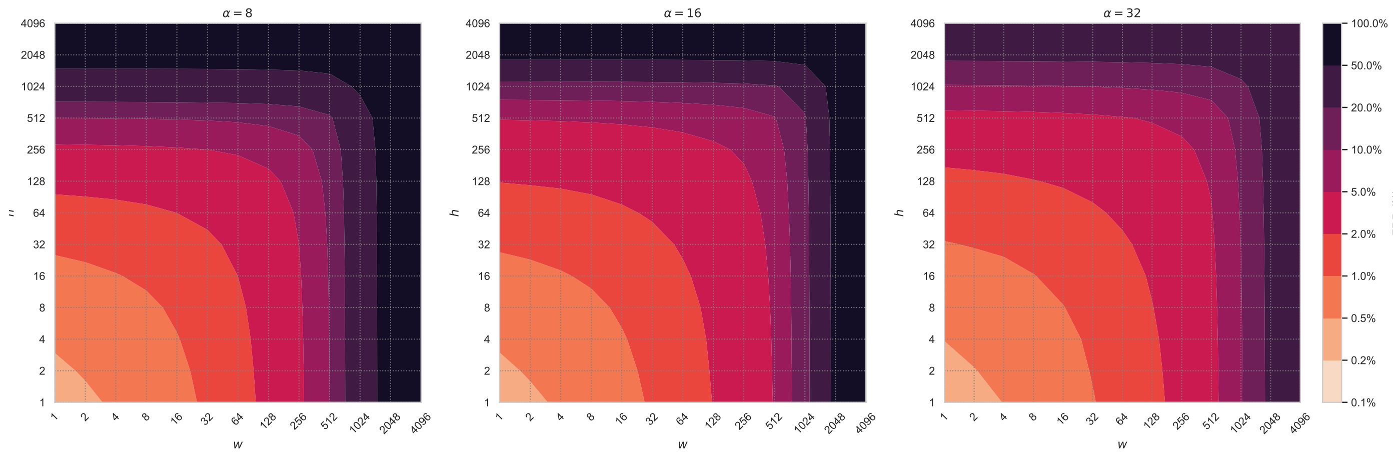


Figure 5.10: Face identification precision of GROTE based on 2D score matrix dimensions h and w, as well as the exponent α.

shown in Table 5.4. We observe first-hand how the lower multiplication depth d requirements of GROTE allow for noticeably faster CKKS operations. This, linked to the drop in latency thanks to the reduction in the number of comparisons (from $T = 2$ to $\lceil 2(h + w)/n \rceil = 1$), yields a significant speedup in the entire system while yielding low error rates for selected $w = 128$, $h = 256$ and $\alpha = 16$ ($FRR \leq 5\%$ as per Figure 5.10, center). We also observe that the latency of the argmax & validation is more than a third of the latency of the entire system, thus a performant solution should sacrifice it at the expense of some loss of practical privacy (due to the increased input leakage). Overall, GROTE is able to reduce the latency of the face identification system by at least 33% (a factor of 1.5), a reduction that would only become more significant for $T > 2$, from K to $2\sqrt{K}$ elementwise comparisons.

Table 5.4: Latency for single-core execution of the listed algorithms. We set $K = n = 16384$ so that the naïve face identif. algorithm requires $T = \lceil 2K/n \rceil = 2$ comparisons, with template elements of $l = 128$ bits, group testing with dimensions $w = 256$ and $h = 128$ ($FRR \leq 5\%$ as per Figure 5.10, center) and exponent $\alpha = 16$, and $\texttt{depth} = 11$ for the comparison polynomial approximation.

Algorithm	Composed of	naïve Latency (ms)	Grote Latency (ms)	Total #mults (d)
CKKS.encrypt	-	99	122	-
CKKS.add	-	1.6	2	-
CKKS.add_plain	-	0.7	0.9	-
CKKS.mult	-	12.4	23	1
CKKS.mult_plain	-	5.7	15.5	1
CKKS.rotate	-	290	438	-
CKKS.relinearize	-	288	443	-
CKKS.mod_switch		9	11	-
Matching ($l = 128$)	$(mult + relin + modswitch) + \log_2(l) * (rotate + add)$	2351	3557	1
Grote ($\alpha = 16, w = 128, h = 256$)	$\log_2(\alpha) * (mult + relin + modswitch) + \log_2(\max(w, h)) * (rotate + add)$	3570	-	4
optMinimaxComp ($\texttt{depth} = 11$)	$\texttt{depth} * (mult + relin + modswitch + add\_plain)$	3433	5202	11
ArgMax	$(mult\_plain + relin + modswitch) + \log_2(\max(w, h)) * (rotate + add)$	2636	3990	1
Validation	$\log_2(\max(w, h)) * (rotate + add) + (mult + relin + modswitch)$	2642	3997	1
Face Identif. (no Argmax)	matching + grote$_?$ + optMinimaxComp$*T_?$	9354	13961	naïve: 23; Grote: 16
Face Identif. (Argmax & valid.)	Face Identif. + argmax + valid	14632	21948	naïve: 25; Grote: 18

5.4.5 Related Works

The core idea of secure face biometrics has been extensively studied before with security guarantees stemming from various privacy-preserving techniques.

Sadeghi et. al. [215] combined homomorphic encryption with garbled circuits for a 2PC privacy-preserving face identification solution using *eigenfaces* [236]. SCiFI [188] employed additively homomorphic encryption and Oblivious Transfer to protect a semi-deterministic region-based face identification system. More recently, Osorio et. al. [189] employed a two-stage face identification consisting of a product quantization-based hashing stage to shortlist some candidates and a reduced homomorphic matching stage based on BFV. Face authentication/verification has also been extensively studied, yielding results fast enough to be used in practice based on the BFV scheme [28] and on other homomorphic encryption schemes [156]. Secure biometric identification has also been proposed for other types of biometric data such as iris recognition [155].

In other line of works, FHE has been widely studied as a technique for privacy-preserving biometrics, from the HE-based biometric access control system of [179], to the packing technique of [245], or [230] showing a clever encoding using packing to perform a biometric matching with one single homomorphic multiplication. [20] used Homomorphic Encryption for fingerprint biometrics, whereas [92] employed both CKKS and BFV for face identification, and [118] proposed the protection of a multi-biometric system. There are other previous works studying secure biometrics, covered in the MPC-based survey from [99] and a collection of FHE-based solutions surveyed in [212].

Lastly, the idea of group testing has touched the field of cryptography before, from pure combinatoric studies [74] to digital fingerprinting and key distribution patterns [227].

5.4.6 Conclusion

This section proposed GROTE, a new algorithm to perform privacy-preserving face identification based on the notion of group testing, and applied it to a solution using the Cheon-Kim-Kim-Song (CKKS) homomorphic encryption scheme. Securely computing the closest reference template to a given live template requires K comparisons, as many as there are identities in a biometric database. GROTE replaces element-wise testing for the K elements of a database by group testing to notably reduce the number of non-linear operations in the encrypted domain from K to up to $2\sqrt{K}$. More specifically, we approximate the max of the coordinates of a large vector by its α-norm (raising to the α-th power and cumulative sum) in a 2D layout, incurring a small impact in the accuracy of the system (CH1) while greatly speeding up its execution (CH2). We implemented GROTE and showed it to be at least 30% more performant than its naïve equivalent for sufficiently large databases $K > 8192$.

5.5 Summary and Concluding Remarks

This chapter dealt with the topic of preserving the privacy of biometric verification, showcasing three novel contributions:

- BIOMFETRICS, a FHIPE solution optimized for online latency that allows for the secure computation of scalar products between templates.
- FUNSHADE, a new 2PC protocol based on a combination of ΠSS and FSS to securely compute various distance metrics and perform threshold comparison.
- GROTE, a new algorithm for privacy-preserving face identification based on group testing to reduce the number of non-linear operations in the encrypted domain while maintaining a high degree of accuracy, applying it to CKKS-based identification.

These works are implemented (CH5) and thoroughly tested with real-world face data, showing that they are efficient (R6), they accurately identify users by their face templates (R5), and they provide suitable privacy guarantees to protect the biometric templates (R3) while enabling biometric authentication/identification. Throughout the chapter we have also studied various sources of accuracy losses (CH1), from the fixed-point encoding in BIOMFETRICS to the approximation of the max in GROTE, and we showed that they can be mitigated by carefully choosing the parameters of these algorithms/protocols.

We have covered multiple ways to preserve the privacy of biometric templates throughout the entire biometric verification process, from the enrollment to the authentication/identification. However, all the underlying techniques (MPC, FHE, FE) neglect the potential leaking of information about the biometric templates in the output revealed at the very end of their operations. In the next chapter we will study the leakage in the output of the verification phase and how it can lead to an unintended recovery of the biometric templates. Since this would break the irreversibility (R2) of our system, we will analyze it and propose strategies to mitigate it as much as possible.

Chapter 6

Revealing the Verification Result

The security provided by our constructions from Chapter 5, and that of most of the privacy-preserving techniques in general (MPC, FHE, FE), does not prevent the reconstructed output $o = f(\boldsymbol{x}, \boldsymbol{y})$ from revealing information about the inputs $\boldsymbol{x}, \boldsymbol{y}$. Indeed, P_{res}, and more generally any entity with access to the result, can leverage on knowledge about the function being computed and attempt to extract information about the inputs from the known outputs by inverting the function being computed $Leak(\boldsymbol{x}, \boldsymbol{y}) \leftarrow Leak(f^{-1}(o))$. This leakage is an open challenge (CH4) often overlooked in the literature, and is particularly relevant in the context of biometric verification.

We dedicate this chapter to study this input leakage and design suitable countermeasures. We formalize input leakage for the context of biometric verification in Section 6.1 and address its impact on the techniques proposed throughout Chapter 5, dedicating Section 6.2 to design a novel solution aiming to mitigate this leakage for BFV-based schemes.

6.1 On input leakage in privacy-preserving biometrics

The output of a private computation protocol always leaks some information about the input, as this leakage is inherent to the mathematical function, independently of the chosen private computation technology. In the domain of Machine Learning, a ML model being used to perform inference over multiple data points can be subject to a wide array of attacks by an adversary receiving the inference results. Some examples of this are:

- *Model extraction attacks* [235, 192] consist of extracting the trained parameters of a deep learning model from a deployed system. These parameters (the model's weights) assemble a new, identical model that can be used for malicious purposes. Alternatively, the attacker can leverage these inference results to train a substitute model in order to steal the functionality of the target model.

- *Model inversion attacks* [103] are characterized by an attacker using a deployed model's outputs to recreate a close approximation of the input data. For example, an attacker could use a deployed face recognition model to reconstruct a person's face from the model's output, potentially allowing them to create a synthetic image of the person.

- *Membership inference attacks* [221] aim to determine whether a specific individual's data was used during the training of a machine learning model, by observing the model's output on a set of queries. This type of attack is a privacy concern for machine learning models that are trained on sensitive personal data, such as medical records, financial transactions or biometric data. An example of a membership inference attack is an attacker attempting to determine whether a specific individual's medical records were used to train a machine learning model that is used to predict the likelihood of a disease. The attacker could make queries to the deployed model using different combinations of the individual's medical data and observe the model's output. Based on the output, the attacker could infer whether the individual's data was used in the training of the model.

Model extraction attacks can be mitigated by using techniques such as model compression, obfuscation, and watermarking [145], as well as by implementing proper security measures in the deployment environment. Mitigating model inversion and membership inference attacks can be achieved by mechanisms such as *differential privacy* [2], which add noise to the model's output to provide a guarantee that the model's output does not reveal any information about the input data.

In the case of biometric systems, a CNN-based feature extractor can be subject to these attacks for which specific mitigation techniques are required (e.g., Adding differential privacy [54]). More importantly, modelling the Biometric Verification as a shallow DL model (Section 2.1.5) allows us to find equivalent attacks in our domain: Model extraction can be seen as an attempt to retrieve the reference templates from the verification results, model inversion as an attempt to steal the live template, and membership inference as an attempt to determine whether a live template belongs to the reference template database. Determining whether a live template belongs to the reference template DB overlaps directly with the goal of a biometric system, hence these attacks are not applicable to our domain[1]. While stealing a live template may raise privacy concerns (as it could break irreversibility (R2)), their volatile nature makes them less of a concern than the reference templates; after all, each live template is supposed to be used for a single verification, whereas the reference templates are fixed and employed for multiple verification attempts. As such, we focus on the unintended reconstruction of the reference templates given multiple verification results, which we call *reference template extraction attacks*.

Model extraction attacks are harder and more costly to execute the more complex the model is. The biometric verification process is crucially vulnerable to these, composed as it is of a linear function (a distance metric calculation) and a non-linear function (threshold comparison, optionally including an $\arg\max$), and thus invertible with relative ease. Centered around biometric verification, and in line with previous works in the topic [56, 170], we formalize input leakage as:

Definition 5 (Input Leakage)**.** *For a biometric verification function* $f_{verif}(f_{dist}, bmx, bmy) : \mathbb{Z}_L^T \times \mathbb{Z}_L^T \mapsto \mathbb{Z}_Q$*, a distance metric* $f_{dist}(bmx, bmy) : \mathbb{Z}_L^T \times \mathbb{Z}_L^T \mapsto \mathbb{Z}_L$ *and two vectors*

[1]One can consider Biometric Verification as a legitimate Membership Inference test. Hence, the system is specifically designed for this task.

$x, y \in bmZ_L^T$, we define input leakage ι as the inverse of the number of calls Φ required to unequivocally determine an unknown input bmy from chosen inputs bmx_i and known outputs $o_i = f_{verif}(f_{dist}, bmx_i, bmy) \quad \forall i \in \{1, \dots, p\}$. As an extension, for n inner product calls we define accumulated input leakage to be $\breve{\iota} = n \cdot \iota$. For $\breve{\iota} \geq 1$, the unknown input bmy can be correctly reconstructed.

We can use **information theory** to model this notion of input leakage:

- *Entropy of hidden input $\boldsymbol{y}$.* The maximum amount of possible entropy present in the unknown input $bmy \in bmZ_L^T$ is $H(\boldsymbol{y}) = T \log_2 L$ bits, considering the extreme case where all of the elements in bmy are independent and identically distributed (i.i.d.) following a uniformly random distribution $\mathcal{U}_{\mathbb{Z}_L}$[2].

- *Entropy of known output o_i.* Similarly to the case above, the maximum possible entropy of the output o_i is $H(o_i) = \log_2 Q$. For n i.i.d. outputs $\{z_1, \dots, z_n\}$, the maximal possible entropy of the outputs is $H(z_1, \dots, z_n) = n \cdot H(o_i)$. This corresponds to the case where the chosen inputs $\boldsymbol{x}_i$ are carefully chosen to yield n maximally independent outputs o_i.

- *Conditional entropy $H(\boldsymbol{y}|\forall o_i)$.* The conditional entropy[3]of the hidden input bmy given the output o_i is $H(\boldsymbol{y}|o_i) = H(\boldsymbol{y}) - H(o_i)$. For the optimal choice of n inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ yielding n independent outputs, the conditional entropy corresponds to $H(\boldsymbol{y}|z_1, \dots, z_n) = H(\boldsymbol{y}) - n \cdot H(o_i)$.

- *Number of calls Φ to f_{verif}.* In line with the definitions above, Φ corresponds to the value of n where there is no uncertainty left and the conditional entropy $H(\boldsymbol{y}|\forall o_i)$ reaches zero, which yields $\Phi = H(\boldsymbol{y})/H(o_i)$.

- *Input Leakage ι.* Correspondingly, we obtain a formulation of the input leakage as $\iota = 1/\Phi = H(o_i)/H(\boldsymbol{y}) = \log_2 Q / T \log_2 L = \frac{\log_L Q}{T}$.

Based on this formulation, we draw some corollaries: The bigger the input space (number of elements per vector T, size of each element L), the lower the leakage, as the attacker needs to perform more calls to determine the unknown $\boldsymbol{y}$. Similarly, the more information given by the output, the higher the leakage:

- A *"match"/"reject" binary output* (Equation 2.4), with $o_i \in \mathbb{Z}_2$ as output space, presents an input leakage of $\iota = 1/(T \cdot \log_2(L))$.

- A *"similarity score" output* (Equation 2.5) $o_i = f_{dist}(\boldsymbol{x}_i, \boldsymbol{y})$, with $o_i \in \mathbb{Z}_L$, leads to an input leakage of $\iota = 1/T$.

[3]In reality the elements in bmy are not i.i.d., as there is some inherent correlation given by the feature extractor to draw templates from the same identity close and push other templates far with respect to f_{dist}, and this correlation could be known to the entity attempting to retrieve $\boldsymbol{y}$. A short discussion on the repercussions of this can be found in Appendix A.4.

[3]We recall that the conditional entropy quantifies the amount of information needed to describe the outcome of a random variable $\mathcal{Y}$ given that the value of another random variable $\mathcal{X}$ is known.

Consequently, for a fixed size of input space given by the feature extractor, the most straightforward method to reduce input leakage in biometric systems is to output the least information possible. For applications like 1:1 biometric authentication, one-bit outputs suffice to determine whether there is a match or not, and hence performing the comparison in a privacy-preserving manner reduces the input leakage of the construction considerably. As such, FHE and FE-based solutions without privacy-preserving comparison to a threshold are more risky to apply in real-world scenarios than threshold-enabled solutions that MPC offers out of the shelf.

This formulation can be extended to $1 : K$ biometric identification for a hidden input $\boldsymbol{Y} \in \mathbb{Z}_L^{K \times T}$ being a set of K templates $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K\}$. Once again, we can study the input leakage for different types of outputs:

- A *"match/reject" binary output* (Equation 2.4), yielding an input leakage of $\iota = 1/(T \cdot K \cdot \log_2(L))$.
- An *"ID index" output* (Equation 2.7), with $o_i \in \mathbb{Z}_{\log_2 K}$ as output space, presents an input leakage of $\iota = 1/(T \cdot K \cdot \log_K(L))$.
- A *"similarity score of max" output* (Equation 2.6), where $o_i = \max\{f_{dist}(\boldsymbol{x}_i, \boldsymbol{Y}^{(k)})\ \forall k\} \in \mathbb{Z}_L$, leading to an input leakage of $\iota = \log_2(K)/(T \cdot K)$.

Equipped with this notion of input leakage, we proceed to analyze its impact on all three schemes proposed in Chapter 5 (BiomFEtrics, Funshade and Grote), to subsequently present a novel protocol named Colmade that minimizes this leakage in a privacy-preserving manner.

6.1.1 Protecting BiomFEtrics against input leakage

The FHIPE scheme we employ in BiomFEtrics (Section 5.2), and by extension all IPE and other privacy-preserving schemes outputting the matching scores $o_i = f_{dist}(\boldsymbol{x}_i, \boldsymbol{y}) = IP(\boldsymbol{x}_i, \boldsymbol{y})$ directly, suffers from input leakage of $\iota = 1/T$, for input vectors of T elements. This leakage can be exploited by an attacker in several ways, presented in the following subsections.

6.1.1.1 Reference template extraction attack

An attacker can make black-box use of the biometric system by submitting $T = \Phi$ live templates chosen[4]to be linearly independent $\boldsymbol{x}_i$, and observing their corresponding outputs $o_i = \{f_{dist}(\boldsymbol{x}_i, \boldsymbol{Y}^{(k)})\ \forall k\}$. We visualize this attack in Figure 6.1.

The attacker can leverage the input leakage to reconstruct each of the hidden reference templates $\boldsymbol{Y}^{(k)}$ by solving the system of T linear equations $\{o_i = \Sigma_{j=0}^{T-1} \boldsymbol{x}_i[j] \cdot \boldsymbol{Y}^{(k)}[j],\ \forall i \in \{1, ..., T\}\}$ for known o_i and $\boldsymbol{x}_i$. Indeed each of the K systems of T linear equations has

[4]We consider an attacker capable of either presenting crafted live biometric traits (e.g., fake faces) to the capture sensor, or controlling the live feature extractor output to be able to submit carefully chosen live templates.

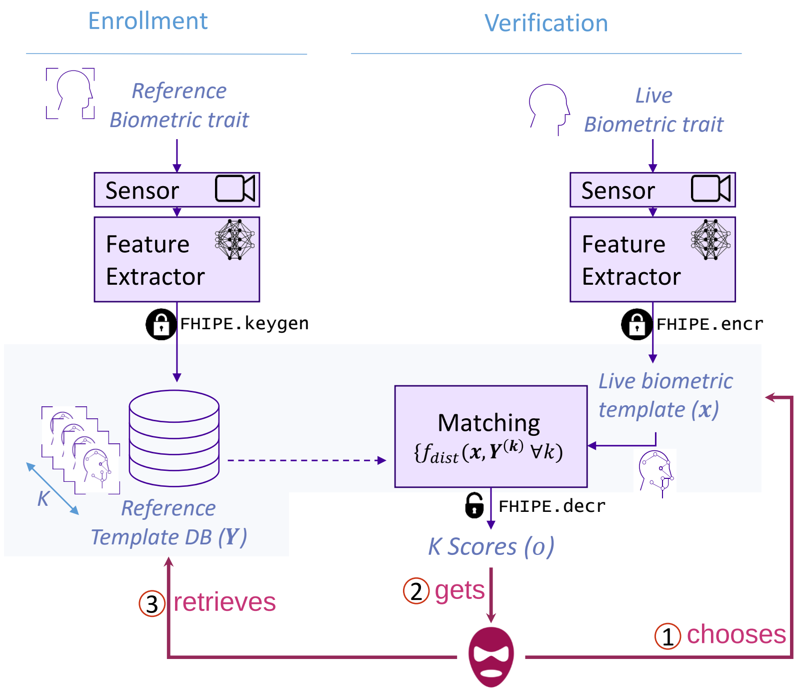


Figure 6.1: Sketch of the reference template recovery attack, where the attacker submits live templates $\boldsymbol{x}_i$, gets the outputs o_i and uses them to extract the reference templates $\boldsymbol{Y}$.

a unique, non-trivial solution for the T unknown variables $bmY^{(k)}[j]\ \forall j$ of each reference template $bmY^{(k)}$ when all the equations are linearly independent.

In our BIOMFETRICS solution (Defined in Section 5.2.3) we propose the countermeasure of limiting the number N of calls to f_{verif} for each reference template to $N < T$, ensuring $\breve{\imath} < 1$. This way, the above system of equations is underdetermined, with $L^{(T-N)}$ possible solutions.

6.1.1.2 Extraction-based brute-force impersonation attack

Even with the limit set above, IPE-based biometric systems answering $N < T$ requests from that same attacker can be subject to brute-force impersonation, where a partially extracted ($\breve{\imath} < 1$) reference template $bm\hat{y}$ is used to impersonate its owner. The attacker first sets the remaining $T - N$ unknown values of $bm\hat{y}$ to arbitrary values and launches a final matching request to attempt a positive matching for a selected identity $k*$ with reference template $bmY^{(k*)}$, such that the IPE result $z = bm\hat{y} \cdot bmY^{(k*)}$ might yield $o \geq \theta$.

To thwart this attack, we set an additional security margin τ to the number N of calls to f_{verif} in our solution, so that $N < (T - \tau)$. Seeking to increase the number of possible solutions of the above system of equations to 2^{80} (80 bits), we fix $\tau \approx 80/\log_2(L)$ for template values in $\mathbb{Z}_L$.

[4]Or to strategically chosen values relying on existing knowledge, as discussed in Appendix A.4.
[4]A proper statistical analysis of the probability of success of this attack whit this limitation in place is left for future work.

6.1.1.3 Limiting the number of requests

Following the strategies suggested above to counter these attacks, we limit the number N of identification requests of the BIOMFETRICS solution to $N < (T - 80/\log_2(L))$, in order to keep $\breve{\imath} < 1$. We enforce this limit via an *access control step* with open instantiation (see the Capture Module in Figure 5.2), which could materialize as an agent-controlled checkpoint or a one-time token generated in the enrollment.

Additionally, we enrich the experiments of BIOMFETRICS by analyzing the verification module in Figure 6.2 based on the number K of identities in our system for a one-time identification scenario. As per all the prior discussion on input leakage, K is limited to strictly less than the template vector length ($K < 128$) to avoid full leakage of the stored templates, and a red area marks the additional 80-bit security margin to thwart brute-force attacks. We observe that the effect of this limit is greater for smaller template vectors, and that employing 6 or more bits in the scheme is sufficient to support $K = 100$ identities[5].

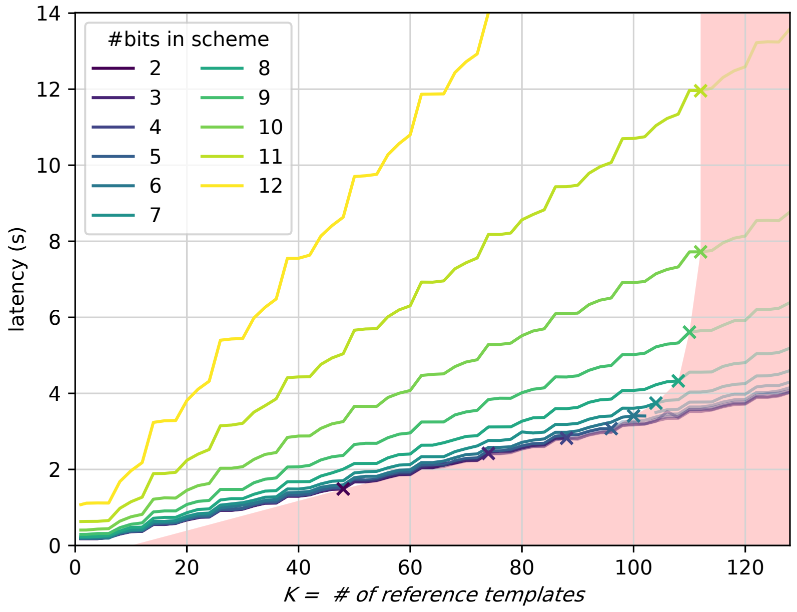


Figure 6.2: Latency (seconds) for the matching module evaluation based on the number of reference templates K and the number of bits l used in the scheme, for input vector elements of size $l = \log_2(L)$ bits.

6.1.2 Input leakage of FUNSHADE and GROTE

As concluded above, it is advisable to output the minimal possible amount of information in the verification phase of a privacy-preserving biometric system. Fortunately, both FUNSHADE and GROTE are designed to output the final matching decision only, and not

[5]Note that the practical scenarios proposed in Figure 5.4 already take these limits into account

the similarity scores: FUNSHADE falls under the category of "match/reject" binary output, whereas the result revealed in GROTE fits the "ID index" type. Thanks to the limited scope of their outputs, we argue both these solutions to be inherently resilient against input leakage.

That being said, the single-party nature of FHE-based biometric systems such as GROTE forces us to trust the correct execution of f_{verif} by the Identification Server (recall Figure 5.8) in order to ensure this resiliency to input leakage[6]. Assuming a semi-honest adversary is non-trivial in practice, as the server could be compromised and the verification function replaced with a malicious one (e.g., a function outputting the elements of a selected reference template). Minimizing the output space size and controlling the decryption of these operations stand out as our best defenses against this threat. Pursuing this idea, we dedicate the next section to the proposal of a novel distributed FHE decryption protocol employing masking to minimize the output space size and thus the risk associated to input leakage (CH4).

6.2 COLMADE: Collaborative Masking in Distributed Decryption for BFV-based Homomorphic Encryption [130]

Abstract. This section proposes a novel collaborative decryption protocol for the Brakerski / Fan-Vercauteren (BFV) homomorphic encryption scheme in a multi-party distributed setting, and puts it to use in designing a leakage-resilient biometric identification solution. Allowing the computation of standard homomorphic operations over encrypted data, our protocol reveals only one least significant bit (LSB) of a scalar/vectorized result resorting to a pool of N parties. By employing additively shared masking, our solution preserves the privacy of all the remaining bits in the result as long as one party remains honest. We formalize the protocol, prove it secure in several adversarial models, implement it on top of the open-source library Lattigo and showcase its applicability as part of a biometric access control scenario.

6.2.1 Introduction

Biometric Identification must rely on secure hardware or trusted parties to hold the personal data vital for their recognition models, and all the biometric data manipulation must follow strict security rules. Hospitals and health specialists are deprived of the advantages of training and using models with all the available data from patients. Banks and finance institutions are limited to the locally available data to prevent fraud and prosecute tax evasion. Child Exploitative Imagery detection models [238] need training data that is in itself illegal to possess.

An observant reader might notice that many of the use-cases just described share two important characteristics: there are multiple parties interested in performing a chosen computation over personal data, and the output of such computation can often be expressed

[6]Note that, contrarily to FHE, MPC protocols such as FUNSHADE can be updated to cover malicious adversaries with existing [77]

with a single bit: Is a user in the database/list of registered users? Does the patient present indicators of cancer? Is a certain entity committing fraud? These aspects will play a key role in the present work.

These technologies alone offer private computation capabilities suited for biometric operations, often based on threat models with *Honest-But-Curious* adversaries, where parties involved perform the selected protocol faithfully and without deviation while attempting to obtain as much information from the private data as possible. Indeed FHE & many MPC techniques fall under this category, whereas some MPC techniques can deal with *Malicious* adversaries (CH3). Besides requiring these stronger threat models, industrial instantiations of these technologies are frequently sought to be auditable, so that an independent auditor may inspect some of the protocol execution based on the public protocol transcript [21].

Even then, the output of a private computation protocol always leaks some information about the input, as this leakage is inherent to the function being computed and independent from the chosen private computation technology (CH4). E.g., model extraction attacks [235, 192] and membership inference attacks [221] in privacy-preserving machine learning inference, or reference template extraction and extraction-based brute-force impersonation attacks covered in Section 6.1. As we discussed before, the most straightforward way to thwart these attacks is to limit the output to yield strictly minimal information (e.g., one bit for binary decisions).

In designing a secure biometric identification system, all these properties are clearly desired: minimal output leakage, auditability and guaranteed data privacy (R1) in stronger threat models. The main motivation of this work is to combine them all, improving on previously proposed biometric systems that only tackle a subset of these properties.

Our Contributions. We propose a novel decryption protocol with collaborative masking based on the multi-party variant [186] of the Brakerski-Fan-Vercauteren (BFV) [100] Homomorphic Encryption scheme. Our protocol makes use of predefined pools of users to perform a decryption in a distributed setting, where each user masks a fragment of the ciphertext during decryption while remaining agnostic of the full computation. We guarantee the privacy of all but one bit of the disclosed output in diverse threat models. Colmade effectively reduces the input leakage to the minimum possible by masking all but the Least Significant Bit of the encrypted output produced by a HE-based private computation. We display its relevance by using it to construct an auditable privacy-preserving biometric identification system. Lastly, we open-source an implementation of this protocol on top of the Lattigo HE library [96].

A fitting application. We target biometric access control for groups where multiple users are expected to get together right before identifying themselves to request access. Together with classic biometric access control scenarios such as airplane boarding or multitudinous events (popular sport matches or concerts), we consider events such as entering a museum, a temporary exhibition, a fair or a semi-professional sports competition. In these selected applications, the computation to perform the biometric identification of the user Alice would happen in the FHE domain, and a set of users (who may or may not include Alice) would collaborate with the gatekeeper to decrypt only the one bit of vital information required to answer "Is Alice in the list of registered people?".

COLMADE is arranged as follows. Section 6.2.2 introduces the single-party & multi-party BFV decryption algorithms. Section 6.2.3 details our design of a biometric access control solution, and leans on it to formalize three protocols: a simplified single-party masked decryption, our flagship privacy-preserving multi-party decryption protocol already displayed in the biometric system, and an extended protocol to provide abort against a malicious adversary in an honest majority setting. Section 6.2.4 covers an in-depth security analysis of our protocols. This section wraps up with a succinct mention to the implementation in Section 6.2.5, related works in Section 6.2.6 and some takeaways in Section 6.2.7.

6.2.2 Preliminaries

This work is grounded on the BFV encryption scheme (Scheme 3) and its distributed counterpart DBFV (Scheme 4). We now focus on the decryption algorithms, as they constitute the foundation of COLMADE. A reasoning on how to select parameters in practice for these two schemes can be followed in Section 3.4 of [186].

6.2.2.1 Decryption in BFV

The decryption of a ciphertext $\mathbf{c}_t = (c_{t_0}, c_{t_1})$ can be described as a two-step process. The first step takes the secret key to compute a noisy upscaled plaintext in R_q as:

$$[c_{t_0}, +sc_{t_1}]_q = \Delta m + e_{\mathbf{c}_t} \tag{6.1}$$

where $e_{\mathbf{c}}$ is the ciphertext error/noise. In the second step, the message is decoded from this upscaled noisy term in R_q to a plaintext in R_t, by downscaling and rounding:

$$\left[\left\lfloor \frac{t}{q}(\Delta m + e_{\mathbf{c}_t}) \right\rceil\right]_t = [\lfloor m + \alpha t + v \rceil]_t \tag{6.2}$$

where $m \in R_t$, $\alpha \in \mathbb{Z}^N$, $v \in \mathbb{Q}^N$. The correctness of the decryption is preserved as long as the noise residue v in the plaintext space be $\|v\| < 1/2$, which translates into an upper bound to the ciphertext noise term of $e_{ct} < \Delta/2$. This is often achieved by choosing a sufficiently large q.

6.2.2.2 Decryption and Noise in DBFV

The main difference between the single-party $\mathsf{BFV.Decrypt}$ protocol summarized in equations 6.1 and 6.2 and its multi-party counterpart $\mathsf{DBFV.ColDecrypt}$ is that an additional error is introduced to preserve the security of the $\langle c_{1s} \rangle_i$ shares based on the decisional RLWE problem.

The distributed secret-key generation protocol yields a global secret key sk whose coefficients, as a sum of K samples of $\mathcal{R}_s$, add up to a maximum of $\|sk\| \leq K$. As a result of $\mathsf{DBFV.ColPubKeyGen}$, the collective public key cpk contains noise $e_{cpk} = \sum_i^K e_i$, implying that $\|e_{cpk}\| \leq KB$.

Thus, a freshly encrypted ciphertext $\mathbf{c}_m = (c_{m_0}, c_{m_1})$ of a message m under a collective public key cpk will decrypt, following equation 6.1 with the single-party BFV.Decrypt, to $[c_{m_0}, +sc_{m_1}]_q = \Delta m + e_{fresh}$, where $\|e_{fresh}\| \leq B(2NK + 1)$. Thus, the worst-case fresh ciphertext noise is linear in the number K of parties.

Conversely, the freshly encrypted ciphertext $\mathbf{c}_m = (c_{m_0}, c_{m_1})$ of a message m under a single-party public key pk will generate, following equation 6.1 and using the multi-party DBFV.ColDecrypt, a similar error term, which then doubles if both DBFV.ColDecrypt and a collective public key are used.

6.2.3 Our solution: collaborative masked decryption

6.2.3.1 Towards biometric database protection

Firstly, let us refresh the main entities of a biometric system for access control, depicted in Figure 6.3:

- A *Biometric Identity Provider (BIP)*, holding a database of reference biometric templates and executing the identification operations upon reception of a live biometric template.
- A service provider acting as *gatekeeper (Gate)*, in charge of capturing live biometric data of an individual requesting access, submitting it to the BIP and authorizing/-denying him access based on the identification result.
- *Users/individuals*, seeking to access the premise or service.

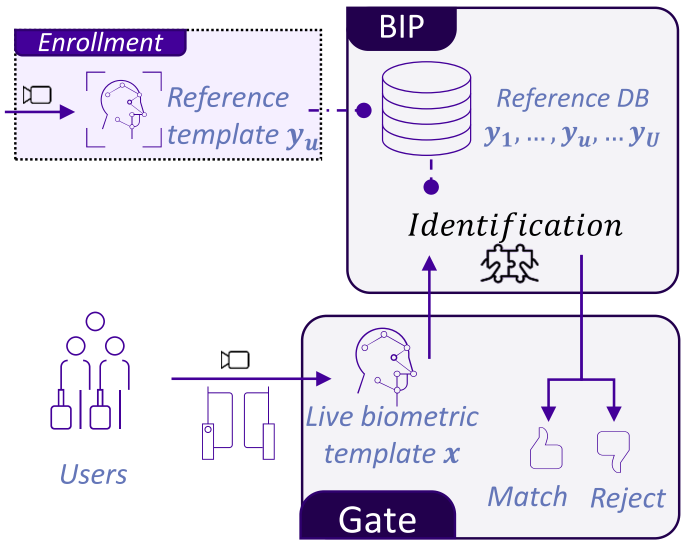


Figure 6.3: Sketch of a biometric system for access control

In biometric systems such as this, the most sensitive asset is the database of stored biometric samples that act as references for identification requests. These *reference templates* could lead to successful impersonation attempts were they to fall in the wrong hands, and their unintended disclosure can lead to severe privacy breaches: knowing who can enter a biometric access system may make these individuals subject of targeted attacks.

To protect the privacy of this database, a straightforward solution is to encrypt it. Enhancing this system with FHE allows the BIP to hold the encrypted database and execute the identification operations over the encrypted domain. Yet, revealing the identification result to the Gate poses several non-trivial concerns:

1. The decryption algorithm makes use of the secret key, but this secret key could also be used to decrypt the database.

2. It is impractical to have one secret key per identity in the database, as individuals might wish to gain access without a personal device holding his key (e.g., smartphone ran out of battery). Also, multi-key homomorphic encryption solutions [175] do not scale well with a high number of keys.

In a real-world instantiation of this biometric system, neither Gate nor the BIP should hold the secret key, as either could team up with the other to fully decrypt the database. Moreover, other issues emerge concerning identification requests formulated by Gate and the responses of BIP:

- The Gate could seek the creation of a False Acceptance (FA) to determine what are the identities present in the database, or search beforehand if a particular person is in there.

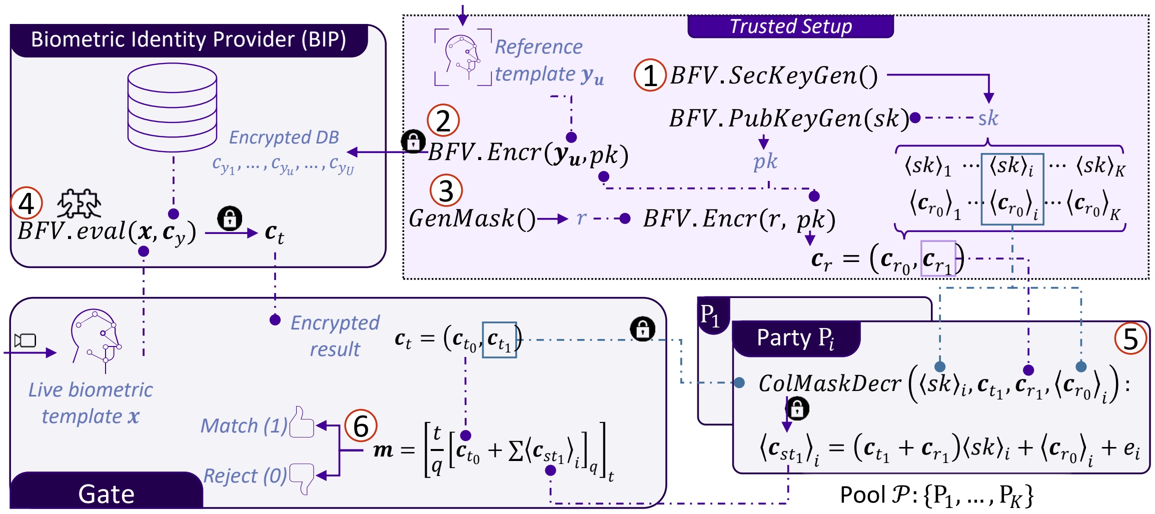
- The BIP could exploit the fact that his answers are encrypted to leak the identities present in the base.

One solution to deal with these issues would be to combine Homomorphic Encryption with Verifiable Computing. However, at the present day this combination is far from being practical [30].

We propose a different approach: splitting the decryption among the users seeking access. We thus hypothesize that a certain number of users will accept to cede a bit of CPU in their smartphones for this task. We could then envision individuals without a phone benefitting from this distributed decryption service thanks to other users. This way, the notion of distributing personal secret keys based on the principle of consent is replaced by a cooperative decryption carried out by those seeking to utilize the identification service. To entice users to do so, we limit their interactions to be only with the Gate, and not with other individuals. [7]

Under this setting the Gate is forced to communicate with the users to decrypt the results of the identification, producing a public communication transcript that will render the service auditable and consequently reduce the risks associated to his requests.

[7]Foreseeing some unwillingness on the user side to participate in this protocol in the real world, we suggest to encourage participation using incentives (e.g., discounts, benefits) to overcome their reluctance.



Biometric Identity Provider (BIP)
Encrypted DB
$c_{y_1}, \ldots, c_{y_u}, \ldots, c_{y_U}$
4
$BFV.eval(\boldsymbol{x}, \boldsymbol{c}_y)$
$\boldsymbol{c}_t$
Trusted Setup
Reference template $\boldsymbol{y}_u$
1 $BFV.SecKeyGen()$
2 $BFV.PubKeyGen(sk)$
sk
$BFV.Encr(\boldsymbol{y}_u, pk)$
pk
3 $GenMask()$
r
$BFV.Encr(r, pk)$
$\langle sk \rangle_1 \cdots \langle sk \rangle_i \cdots \langle sk \rangle_K$
$\langle \boldsymbol{c}_{r_0} \rangle_1 \cdots \langle \boldsymbol{c}_{r_0} \rangle_i \cdots \langle \boldsymbol{c}_{r_0} \rangle_K$
$\boldsymbol{c}_r = (\boldsymbol{c}_{r_0}, \boldsymbol{c}_{r_1})$
Live biometric template $\boldsymbol{x}$
Encrypted result
$\boldsymbol{c}_t = (\boldsymbol{c}_{t_0}, \boldsymbol{c}_{t_1})$
Match (1)
Reject (0)
6
$\boldsymbol{m} = \left[\frac{t}{q} \left[\boldsymbol{c}_{t_0} + \Sigma \langle \boldsymbol{c}_{st_1} \rangle_i \right]_q \right]_t$
Gate
P_1
Party P_i
5
$ColMaskDecr\left(\langle sk \rangle_i, \boldsymbol{c}_{t_1}, \boldsymbol{c}_{r_1}, \langle \boldsymbol{c}_{r_0} \rangle_i\right):$
$\langle \boldsymbol{c}_{st_1} \rangle_i = (\boldsymbol{c}_{t_1} + \boldsymbol{c}_{r_1}) \langle sk \rangle_i + \langle \boldsymbol{c}_{r_0} \rangle_i + e_i$
Pool $\mathcal{P}: \{P_1, \ldots, P_K\}$

Figure 6.4: System diagram of a biometric access control using the COLMADE protocol for single-bit masked results.

6.2.3.2 COLMADE for group biometric identification

Motivated by the use-case of biometric access control for group events, we display the COLMADE protocol in Figure 6.4, to answer if user Alice is allowed to access the museum. We distinguish three types of actors in our scenario:

- *BIP*, holding the encrypted database of reference templates belonging to users allowed to access.
- *Gate*, in charge of capturing the live biometric template of the users requesting access, submitting it to the BIP (possibly encrypted), receiving the encrypted result and aggregating the decryption shares.
- $\mathsf{P}_1 \ldots \mathsf{P}_i \ldots \mathsf{P}_K$. Pool of users holding shares of the global secret key and global masking polynomials. They collaborate to perform a masked decryption of the encrypted identification computation. Alice might or might not be among them. We contemplate multiple pools of users, each with their own sharing of the same global secret key. K needs not be fixed equally for all the pools of users.

This protocol requires a trusted setup to function. Nevertheless, most of these setup operations could be instantiated with alternative protocols that do not require full trust.

Armed with these distinctions, the system would carry out Alice's identification following this steps:

1. **Key Generation**: The trusted setup generates a global secret and public BFV keys and secret-shares the secret key in K shares. A distributed alternative would be to instead employ DBFV.SecKeyGen and DBFV.ColPubKeyGen for key generation, and then use the Enc2Share protocol from [186].

2. **Reference Database Encryption**: During the enrollment process, the users to be included in the access list would yield biometric templates to act as references once they arrive at the gate. These reference templates are encrypted with the public key generated in the previous step, and are then sent to the BIP for safe storing.

3. **Randomness generation** Following the setup of Protocol 18, the masking polynomial is sampled, encrypted and secret shared.The trusted entity then sends all the required pieces to each of the parties enrolling. In practice this would typically happen in a previous "offline" phase, after the enrollment, that could alternatively be based on correlated randomness [140].

4. **Encrypted identification** Once Alice approaches the gate, a biometric template is extracted from her by the Gate and sent to the BIP. This live template could be encrypted at the cost of slower operations in the BIP, but guaranteeing privacy of the live template in the BIP. Once received, the BIP would perform the encrypted identification (e.g., vector-matrix multiplication followed by comparison to threshold [138] and aggregation) with the encrypted DB, sending the encrypted result back to the Gate.

5. **Collaborative Masked Decryption** Upon receiving the encrypted result, the Gate would request a decryption to a pool of users by sending them all the second polynomial of the encrypted result. Each of the parties answers back with a share of that decrypted and masked polynomial.

6. **Result** The Gate aggregates all the decrypted polynomial shares with the first polynomial of the encrypted result, decrypts and decodes the underlying message and answers, based on the single LSB bit disclosed, if Alice is in the list and can access the premises/service.

6.2.3.3 Masked Decryption

The goal of the COLMADE protocol is to conceal all bits save a single LSB of the underlying message during BFV decryption. To that end we add a masking term as part of the decryption protocol. Portrayed in Figure 6.5 as an additive term in the plaintext ring, this mask will depend on the type of encoding being used:

- *Base encoding* places one base-b digit per polynomial coefficient. We distinguish two cases:
  - For an even b, the LSB of the underlying integer value depends only of the polynomial coefficient $j = 0$, corresponding to the b^0 term. The desired masking term r would have to fully hide all coefficients except $j = 0$, and this coefficient ought to have all but the LSB bit masked.
  - For odd b, the LSB depends on all the coefficients of the polynomial, and thus no suitable additive mask can be applied. We disregard this case from this point onward.

- *Packed encoding* places one vector element per polynomial coefficient. The desired mask r would have to completely conceal all coefficients but one, and that coefficient should have all its bits obscured except the LSB.

Protocol 17 MaskDecrypt($sk, \mathbf{c}_t$)$\rightarrow$ m_{LSB}

```
LET $sk = s$ a secret key, $\mathbf{c} = (c_0, c_1)$ a ciphertext.
SAMPLE $r \leftarrow \mathcal{R}_{[R_t]}$
COMPUTE $m_{LSB_q} = [c_0 + sc_1 + \Delta r]_q$
OUTPUT $m_{LSB} = \left[\lfloor m_{LSB_q} * t/q \rceil\right]_t$
```

Notice how the mask for packed encoding and the mask for even-based encoding would look very similar. For our desired masking polynomial r, we draw $N-1$ coefficients as $r[j] \sim \mathcal{U}\left(\mathbb{Z}_{[0,t)}\right) \ \forall j \neq 0$, and one single coefficient from $r[0] \sim 2\mathcal{U}\left(\mathbb{Z}_{[0,t/2)}\right)$ to preserve the LSB.[8] We define this mask distribution as $\mathcal{R}_{[R_t]}$.

If we introduce this mask r in the right hand side of Equation 6.2, we would achieve our desired functionality. To balance the equation, we add Δr in the other side.

$$\begin{gathered} r \leftarrow \mathcal{R}_{[R_t]} \quad m_{LSB} = [m + (r)]_t \\ LSB = (Decode(m_{LSB})[0]) \bmod 2 \end{gathered} \tag{6.3}$$

$$\left[\left\lfloor \frac{t}{q}\left(\Delta(m+r) + e_{c_t}\right)\right\rceil\right]_t = [\lfloor (m+r) + at + v \rceil]_t \tag{6.4}$$

Since $\|\Delta r\| < q$, we can introduce this mask in Equation 6.1:

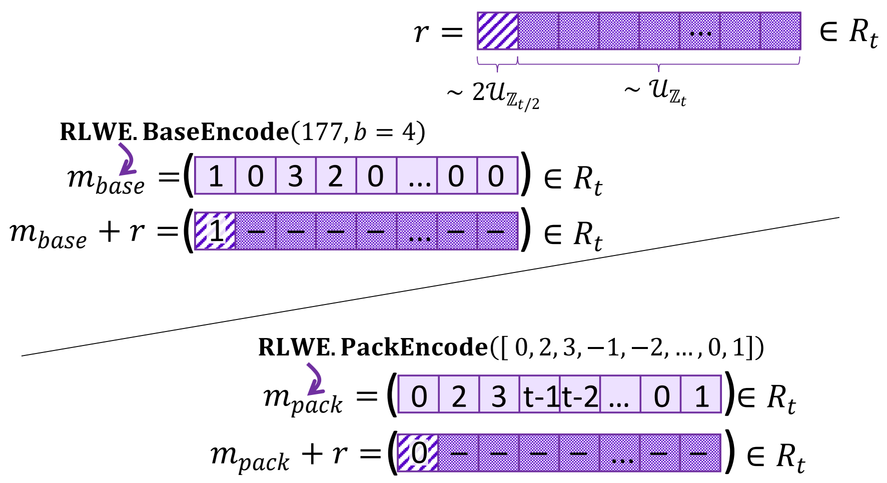
$$[c_0 + sc_1 + \Delta r]_q = \Delta m + e_{c_t} + \Delta r \tag{6.5}$$

We remark that adding masking might cause $m[0] + r[0] \geq t$, in which case the modulo operation would kick off and flip the LSB (recall that t is prime and thus an odd number). The following limitation is imposed for the coefficient $j = 0$ containing the LSB[9]:

$$m[0] + r[0] < t \tag{6.6}$$

This limitation effectively imposes $m[0] \in \{0, 1\}$ to avoid the LSB flip and preserve correctness. To overcome it, the ($\mathsf{mod}\ t$) operation could be approximated by ($\mathsf{mod}\ (t-1)$) with $(t-1)$ being an even number, ensuring the LSB preservation after applying modular reduction. Translated into the R_q domain, $\mathsf{mod}\ q$ reductions during and after decryption would then be approximated by $\mathsf{mod}\ q'$ with $q' = \Delta * (t-1) = q - \Delta$.

[8]In the packed encoding we could chose to preserve the LSB of an arbitrarily chosen coefficient j. We set $j = 0$ for convenience.



$r =$ … $\in R_t$
$\sim 2\mathcal{U}_{\mathbb{Z}_{t/2}}$
$\sim \mathcal{U}_{\mathbb{Z}_t}$
RLWE. BaseEncode(177, $b = 4$)
m_{base} = (1 0 3 2 0 … 0 0) $\in R_t$
$m_{base} + r$ = (1 – – – – … – –) $\in R_t$
RLWE. PackEncode([0, 2, 3, −1, −2, … , 0, 1])
m_{pack} = (0 2 3 t-1 t-2 … 0 1) $\in R_t$
$m_{pack} + r$ = (0 – – – – … – –) $\in R_t$

Figure 6.5: Visualization of masking in the plaintext domain for the arbitrary encodings of Fig. 3.1

6.2.3.4 Collaborative Masked Decryption

Extending the masked decryption to a multi-party collaborative setting requires merging Protocol 17 with DBFV.ColDecrypt. To do so we require the sampling of $r \leftarrow \mathcal{R}_{[R_t]}$ to be handed to the different parties in the form of shares $\langle r \rangle_i$. In addition, we encrypt the mask shares. For convenience and performance, we swap the order, first encrypting the global masking polynomial $\mathbf{c}_r = \mathsf{BFV.Encrypt}(r, pk)$ and then splitting it into shares in the encrypted domain by adding encoded secret shares of zero.

Thanks to the standard properties of arithmetic secret sharing [220], you require the results of all the K parties to reconstruct the correct masking polynomial r, and any sum of shares from less than K parties $\sum_i^{<K} \langle r \rangle_i$ is indistinguishable from a uniformly random sampling $r' \leftarrow \mathcal{U}(R_q)$.

The full ColMaDe decryption is outlined in Protocol 18.

Protocol 18 ColMaskDecr(c, $\langle sk \rangle_1, \ldots \langle sk \rangle_K$) $\rightarrow m_{LSB}$

Let $\langle sk \rangle_i$ the private share of a global secret key $s = \sum_i^K \langle sk \rangle_i$; $\mathbf{c} = (c_0, c_1)$ a ciphertext; $r \leftarrow \mathcal{R}_{[R_t]}$ a masking polynomial encrypted with $\mathsf{BFV.Encrypt}(r, pk) \rightarrow \mathbf{c}_r = (c_{r_0}, c_{r_1})$ and splitting c_{r_0} in K shares $\langle c_{r_0} \rangle_i$

Setup: P_i holds s_i, $\langle c_{r_0} \rangle_i$ and c_{r_1}.

P_i: Sample $e_i \leftarrow \mathcal{X}_{[R_q]}$

P_i: Compute $\langle c_{1s} \rangle_i = [\langle sk \rangle_i (c_1 + c_{r_1}) + e_i + \langle c_{r_0} \rangle_i]_q$

Any: Output $m_{LSB} = \left[\left\lfloor \frac{t}{q} \left[c_0 + \sum_i^K \langle c_{1s} \rangle_i \right]_q \right\rceil \right]_t$

6.2.3.5 Replicated Masked Decryption

We can customize the Colmade protocol under a malicious setting by sending/setting J shares of the secret key $\langle sk \rangle_i$ and J shares of the first polynomial of the encrypted mask $\langle c_{r_0} \rangle_i$ to each party P_i, in a replicated sharing scheme. Protocol 19 details this modification.

In this replicated setting, each party would compute J shares of $\langle c_{1s} \rangle_i$, whose individual decryptions should yield uniformly random yet equal results for the same input shares of sk and c_{r_0}. By comparing these auxiliary decryptions, the party in charge of aggregating all the results and outputting m can detect up to $J-1$ parties deviating from the protocol, thus allowing the aggregator to *detect* these malicious adversaries and *abort* the decryption. However, this replication technique lowers the number of parties required to reconstruct the entire mask from K to at least $K-J$, thus making this technique suitable only for when a majority of parties is honest. We study the relation between J and the number of malicious corruptions in the pool $|\mathcal{A}_{\mathcal{P}}|$ in section 6.2.4.4.

Protocol 19 ReplColMaskDecr($\langle sk \rangle_1 \dots \langle sk \rangle_K, \mathbf{c}, J) \rightarrow \mathrm{m}_{LSB}$

Let $s_i = \langle sk \rangle_i$ private shares of the global secret key $s = \sum s_i$; $\mathbf{c} = (c_0, c_1)$ a ciphertext; $r \leftarrow \mathcal{R}_{[R_t]}$ a masking polynomial encrypted with $\mathsf{BFV.Encrypt}(r, pk) \rightarrow \mathbf{c}_r = (c_{r_0}, c_{r_1})$ and secret-sharing c_{r_0} in K shares $\langle c_{r_0} \rangle_i$; with $\{i_J\} = \{(i+j)\%K\}$ for $\{0 \cdots j \cdots J-1\}$

Setup: P_i holds J shares $s_{\{i_J\}}$, c_{r_1} and J shares $\langle c_{r_0} \rangle_{\{i_J\}}$.

P_i: Sample J times $e_{\{l\}} \leftarrow \mathcal{X}_{[R_q]}$.

P_i: Compute $\langle c_{1s} \rangle_{\{i_J\}} = \left[s_{\{i_J\}}(c_1 + c_{r_1}) + e_{\{i_J\}} + \langle c_{r_0} \rangle_{\{i_J\}} \right]_q$

Any: Check equality among all $\lfloor t/q \ \langle c_{1s} \rangle_{\{i_J\}} \rceil \ \forall i, j$.
Abort if non equal.
Output $m_{LSB} = \left[\left\lfloor \frac{t}{q} \left(c_0 + \sum_i^k \langle c_{1s} \rangle_i \right)_q \right\rceil \right]_t$

6.2.4 Security Analysis

The Colmade protocol seeks to guarantee the privacy of all bits but a single LSB in a ciphertext. The underlying real-world motivation in the biometrics domain was mentioned already: the most precious resource in our system is the database of reference templates, since they are often tightly linked to the person, and thus non revocable like passwords or tokens. Hence, in scenarios where the reference templates are used for multiple applications, its theft could lead not only to a potential impersonation when accessing the desired service/premise, but to a severe identity theft across applications.

To study the security of our protocol, we generalize that an adversary corrupting the Gate also corrupts the BIP, which means that this combined corruption would grant the adversary full access to the encrypted database and can perform chosen ciphertext attacks (CCA) using the pool of users as decryption oracle. We analyze several threat models:

1. Gate is semi-honest, and up to $K-1$ parties in the pool are semi-honest (at least one honest user per pool).
2. Gate is malicious, and up to $K-1$ parties are semi-honest (at least one honest user per pool).
3. Gate is semi-honest, a minority of parties in the pool are malicious (at least $\lceil K/2 \rceil$ honest user per pool).

6.2.4.1 On Privacy of Colmade

Privacy in the semi-honest pool We first provide a security proof for the proposed Colmade protocol in the standalone passive adversary model for the pool of users, that we base on the *decision RLWE assumption* [180]. We formulate our proof using the ideal/real simulation paradigm [171]: We show that, for every possible adversarial subset $\mathcal{A}$ of all the computing parties in the pool $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_K\}$, there exists a simulator program S that can simulate $\mathcal{A}$'s view in the protocol, when provided only with $\mathcal{A}$'s input and output. To achieve semantic security [113], we require $\mathcal{A}$ not be able to distinguish the simulated view from the real one. Note that the view of the adversary after the setup is the full transcript (public transcript property). For a given value x, we denote $\tilde{x}$ its simulated equivalent. We consider computational indistinguishability between distributions, and denote it as $x \overset{c}{\equiv} \tilde{x}$. We denote $\mathsf{view}^{\mathsf{ColMaskDecr}}$ to the transcript of Protocol 18, consisting of all the shares $\{\langle c_{1s}\rangle_1, \ldots, \langle c_{1s}\rangle_i, \ldots, \langle c_{1s}\rangle_K\}$ of c_{1s} in R_q.

Simulator 1 $S^{\mathsf{ColMaskDecr}}$

Players: The simulator is given $\{\langle sk\rangle_i, \langle c_{r0}\rangle_i, e_i\} \forall i$ and c_{r_1} by the trusted setup.
The simulator receives $\mathbf{c}_{t_1}$ from the Gate.

Output: for each party P_i in the pool:

$$\langle \widetilde{c_{1s}}\rangle_i = \begin{cases} [(c_{t_1} + c_{r_1})\langle sk\rangle_i + e'_i + \langle c_{r0}\rangle_i]_q & \text{if } \mathsf{P}_i \in \mathcal{A} \\ \leftarrow \mathcal{U}(R_q) & \text{if } \mathsf{P}_i \in \mathcal{H} \\ \left[c_{1s} - \sum_{\mathsf{P}_i \in \mathcal{P}\setminus \mathsf{P}_h} \langle \widetilde{c_{1s}}\rangle_i \right]_q & \text{if } \mathsf{P}_i = \mathsf{P}_h \end{cases}$$

Theorem 4. *(ColMaskDecr* privacy in the semi-honest pool model*) For each possible set of corrupted parties $\mathcal{A} \subset \mathcal{P}$ by a passive adversary with $|\mathcal{A}| \le K-1$, there exists a simulator $S^{\mathsf{ColMaskDecr}}$ such that:*

$$S^{\mathsf{ColMaskDecr}} \overset{c}{\equiv} \mathsf{view}^{\mathsf{ColMaskDecr}}$$

Proof. First, Theorem 4 forces at least one arbitrarily chosen party P_h to be honest. We denote $\mathcal{H} \triangleq \mathcal{P}\setminus(\mathcal{A} \cup \{\mathsf{P}_h\})$ to the set of all other honest parties, so that the tuple $(\mathcal{A}, \mathcal{H})$ represent any partition of $\mathcal{P}\setminus\mathsf{P}_h$. We consider the error term e_i sampled as a part of the protocols as private input to the protocol.

We can now consider the distribution of the simulated and real views. The *decision-RLWE* assumption suffices to prove it in the absence of the masking term, as for an adversary that does not know $\langle sk_i \rangle$ nor e'_i, we get that:

$$(\langle sk_i \rangle c_{t_1} + e'_i, c_{t_1}) \overset{c}{\equiv} (a \leftarrow R_q, c_{t_1})$$

The addition of the masking terms only increases the randomness of the first element in the tuple, thus the equation holds true. □

Minimum leakage of the output. When considering the threat model #1 (all semi-honest), we resort to Theorem 4 to show that the protocol itself does not reveal more than what the output does. Now we seek to prove that the output reveals only one bit.

Theorem 5. (*ColMaskDecr* 1-bit leakage of output in the semi-honest pool model) *For each possible set of corrupted parties $\mathcal{A} \subset \mathcal{P}$ by a passive adversary with $|\mathcal{A}| \leq K-1$, the protocol reveals a maximum of one bit from the encrypted message, the LSB of the first coefficient in the underlying encoded polynomial.*

Proof. Since the output $m_{LSB} = [m+r]_t$ is the result of adding a mask r to the underlying message m, and this mask contains uniformly random values in $\mathbb{Z}_{[0,t)}$ for all $j \neq 0$, we get that:

$$\begin{aligned} P(m[j] = a \mid m_{LSB}[j] = b) = & \\ & \frac{P(m[j] = a) \cdot P(m_{LSB}[j] = b | m[j] = a)}{P(r[j] = b)} = \\ & P(m[j] = a) \end{aligned} \tag{6.7}$$

This is because $m_{LSB}[j] \overset{c}{\equiv} r[j] \sim \mathcal{U}(\mathbb{Z}_{[0,t)})$, and thus an adversary receiving $m_{LSB}[j]$ obtains no information beyond what he already knew about $m[j]$, showing how all slots $j \neq 0$ of the message are perfectly masked.

For $j = 0$ we can proceed in a similar fashion to prove that $P(m[0] \bmod 2 = a \mid m_{LSB}[0] \bmod 2 = a) = 1$ with $a \in \{0, 1\}$ if $m[0] \in \{0, 1\}$, which requires the Gate & BIP to follow the protocol as specified (honest or semi-honest) and comply with the limitation from equation 6.6. At the same time, $P(m[0]/2 = b \mid m_{LSB}[0]/2 = c) = P(m[0]/2 = b)$, showing that all but the LSB of $m_{LSB}[0]$ are perfectly masked.

In the event of $m[0] \notin 0, 1$, arising from a malicious Gate & BIP submitting a chosen $\mathbf{c}_t$ (threat model #2), we get that m_{LSB} yields the correct LSB if $(b+r[0]) < t$ with probability $(t-b)/t$, and flips with probability b/t. Interestingly, all the other bits remain perfectly masked, since $P(m[0]/2 = b \mid m_{LSB}[0]/2 = c) = P(m[0]/2 = b)$ in this case too. [10]

Based on this, we can conclude that, with up to $K-1$ semi-honest parties in the pool, our protocol discloses only one bit if the protocol is followed by the Gate & BIP and less than one bit otherwise. □

[10] If using the modulo approximation hinted at the end of section 6.2.3.3, the LSB would be retrieved correctly $\forall\, m[0] \in \mathbb{Z}_{[0,t)}$, covering threat models #1 and #2.

6.2.4.2 On Correctness of Colmade

To ensure that the message requested by the Gate is correctly decrypted in the presence of a minority of malicious users in the pool (threat model #3), we employ the ReplColMaskDecr enhanced protocol described in Section 6.2.3.5. We resort to the replication of shares inside each pool to effectively overcome a small number of malicious users by detecting misbehavior and aborting. Following Protocol 19, a semi-honest Gate in charge of aggregating all the masked shares can detect when two replicas are different in an honest-majority pool (with some additional limitations studied in Section 6.2.4.4), and abort the decryption, potentially requesting the decryption again to a different pool.

Beyond this, the correctness of the biometric system depends on Gate & BIP following the protocol by performing the valid operations for encrypted identification and submitting the encrypted result to a pool for decryption.

6.2.4.3 On well known FHE attacks

Fully Homomorphic Encryption schemes are known to be secure against Chosen Plaintext Attacks (CPA) as a direct consequence of the indistinguishability property that is the base of their security (e.g., the *decision RLWE assumption* for BFV and CKKS schemes). However, many of these schemes fail catastrophically against Chosen Ciphertext Attacks (CCA), and they are all insecure against adaptive CCA (or CCA-2) [174].

Beyond this, an adversary with access to a decryption oracle can, with one single well-chosen query, reveal the entire secret key in the standard BFV scheme [195],being just as applicable to DBFV to extract the global secret key. Since our solution provides a sort of decryption oracle to the Gate, we analyze the impact of this attack on our system.

The attack of [195] is rooted in a malicious adversary corrupting the Gate and performing one decryption request to a decryption oracle:

1. Craft a fake ciphertext $\mathbf{c}_t = (c_{t_0}, c_{t_1}) = (\mathbf{0}, \Delta)$[11]
2. Request decryption of $\mathbf{c}_t$ to the oracle. Following equation 6.1, he obtains in result $p_s = [0 + \Delta * s]_q$, where s is the secret key and $e_{c_t} = 0$.
3. Locally downscale this plaintext using equation 6.2 to obtain an estimation of key: $\hat{s} = p_s/\Delta$.

In the case of Colmade, this attack is much less problematic. Step 2 of the attack would consist of Gate requesting a pool $\mathcal{P}$ to decrypt the result. Using equations 6.3 and 6.4, the result would be $p_s = [0 + \Delta * s + \Delta r]_q$, which in step 3 translates into $\hat{s} = p_s/\Delta + r$. The crucial difference is the mask addition. For all $j \in [1, N-1]$ save the first coefficient $j = 0$, this translates into $s[j] + r[j]$, where $r[j] \sim \mathcal{U}(\mathbb{Z}_{[0,t)})$ acts as a perfect one-time pad over the underlying plaintext space $[0, t)$, completely hiding that secret key coefficient. For $j = 0$ this leaks one bit of $s[0] \sim \mathcal{U}(\{-1, 0, 1\})$, thus $s[0]$ would be recoverable with two queries. Most importantly, the coefficient of s being leaked is always $j = 0$, as the other coefficients

[11]All the coefficients in polynomial c_{t_0} set to 0 and in polynomial c_{t_1} set to Δ.

are perfectly hidden and no rotations are applied as part of the decryption, which would hypothetically help to extract other coefficients.

While CCA attacks are still feasible, our single-bit output coupled with the auditability property makes these attacks[12] much less practical due to the number of malicious requests required.

6.2.4.4 On the choice of users, pools and replicas

In principle the choice of pool for each decryption request of the Gate must be a random choice among all the pools available. We propose the use of a Verifiable Delay Function [32] with a chosen random beacon [72, 35] to guide the choice of pool for every decryption, as a way to have an unbiased choice and facilitate auditability.

The number of users per pool K is left open, knowing that in practice bigger pools are harder to manage (e.g., bigger delay) and more error prone, but also more theoretically secure, since per Theorem 4 it in the real world it increases the chances of including one honest user needed to preserve the privacy of the collaborative decryption. The choice of users for each pool should ideally also be random, and it is included in the trusted setup.

The ReplColMaskDecr protocol can used to address threat model #3. If so, the number of replicas per user J should be set such that the adversary cannot reconstruct the secret key. This leads to:

$$|\mathcal{A}_{\mathcal{P}}| * J < K \tag{6.8}$$

To support a maximum number of malicious users in the pool $|\mathcal{A}_{\mathcal{P}}|$, J should thus be set small. However, to ensure that each replica is at least in the hands of one honest user we require

$$J \geq |\mathcal{A}_{\mathcal{P}}| + 1 \tag{6.9}$$

Hence, to ensure security in an honest majority, we limit the maximum number of malicious corruptions to:

$$|\mathcal{A}_{\mathcal{P}}| < \lfloor \sqrt{K} \rfloor \tag{6.10}$$

6.2.4.5 On Auditability of Colmade

An external auditor could enroll as a user in the pool to take part in the COLMADE decryption protocol, and since the protocol requires communication with all parties, an eavesdropping auditor with access to the public transcript of these communications would know about all decryption requests.

The biometric solution based on Colmade can thus be audited by an external entity with the following items:

- The public transcript of all the communications between Gate and users testify of the number of decryption requests performed. Following 5, the auditor could infer an upper bound on the number of bits extracted from the database if all the decryption requests contained maliciously crafted ciphertexts.

[12]An overview of FHE key recovery attacks can be found in [95].

- Since our solution does contain a trusted setup, an auditor suspecting malfeasance could request access to the secret key material to open some decryption requests.
- For the choice of the pools, a Verifiable Delay Function with some well chosen random beacon could serve as an unbiased yet auditable way for the Gate to select what pool to use for each decryption request.

6.2.5 Implementation

We implement the COLMADE protocol on top of the Lattigo [96] homomorphic encryption library, including examples of usage and correctness checks. Our Golang implementation is open-sourced in `https://github.com/ibarrond/colmade`.

Simply encoding each vector element into an upscaled coefficient in R_q would lead to slower ring operations when $q < 2^{64}$, as it requires arbitrary-precision arithmetic that is much more computationally expensive than standard integer arithmetic in a 64-bit machine. In practice [96], vectors are encoded to polynomials using the Chinese Remainder Theorem (CRT) into a Residue Numeral System (RNS) form [17] by decomposing $q = q_1 \cdot q_2 \cdot \ \ldots \ \cdot q_l$ into coprime factors smaller yet close to machine word size of 2^{64}.

While all our protocols apply in the RNS variant of BFV, it is worth noting that the modulo approximation of Section 6.2.3.3 could be applied to a single factor, $q' = q_1' \cdot q_2 \cdot \ \ldots \ \cdot q_l$ with $q_1' = (q_1 - q_1/\Delta)$.

6.2.6 Related Work

Preceding work employing masking for RLWE instances has been focused on protecting the secret key during decryption operations, to upgrade the Chosen-Plaintext-Attack (CPA) security guarantees of RLWE cryptosystems into Chosen-Ciphertext-Attack (CCA1/CCA2). In this line, additively homomorphic masking was proposed to output a a secret-shared result that will later be reconstructed during decoding [205], and an follow-up work proposed a decryption outputting boolean shares suitable for derivation of a symmetric key to be used during decoding [206]. Further down the line, [187] proposes an adaptation of RLWE schemes to render them CCA2-Secure based on a post-quantum variant of the Fujisaki-Okamoto (FO) transform combined with masked binomial sampling to secure a re-encryption process.

The idea of masking in HE has also been studied previously in the form of slot masking, a method to collapse multiple unique-repeated-value ciphertexts into a single ciphertext for encrypted vector-matrix multiplication: multiply each ciphertext with a mask containing a 1 set in a chosen slot and 0 in all the other slots. We saw this technique applied for HE-based applications in the context of phishing web page classification [70], and in HE-Friendly privacy-preserving mobile neural network architectures [176].

In other line of works, FHE has been widely studied as a technique for privacy-preserving biometrics, from the HE-based biometric access control system of [179], to the packing technique of [245], or [230] showing a clever encoding using packing to perform a biometric matching with one single homomorphic multiplication. [20] used Homomorphic Encryption for fingerprint biometrics, whereas [92] employed both CKKS and BFV for face identification, and [118] proposed the protection of a multi-biometric system.

While there are many previous works studying secure biometrics with MPC [99] and FHE [212], to the best of our knowledge this is the first work to contemplate the intersection of multi-party homomorphic encryption [81, 186] with biometrics. Lastly, while the vanilla DBFV decryption of [186] would already provide auditability and data privacy against a semi-honest adversary, our work extends it to malicious corruptions and yields minimum input leakage thanks to the collaborative masking embedded in the decryption protocol.

6.2.7 Conclusion

COLMADE proposes a novel collaborative masking decryption protocol for the multi-party BFV scheme guaranteeing data privacy, minimal output leakage (1 bit), and auditability. Our protocol makes use of predefined pools of users to perform a decryption in a distributed setting while adding an additively shared encrypted masking term. We showcase its applicability as part of a biometric access control solution where groups of users get together for orderly individual identification. We prove this protocol secure against $K-1$ corruptions of a semi-honest adversary, and show an enhanced version using replicas to be resilient against $\lfloor\sqrt{K}\rfloor$ active corruptions of a malicious adversary. We analyze practical security aspects of the biometric solution, and open-source implementations of these protocols on top of the Lattigo library.

6.3 Summary

We kicked off this chapter with a formal definition of input leakage (CH4), its modeling using information theory and an in-depth breakdown of its consequences for biometric systems. We followed up with an analysis of the input leakage present in the three solutions proposed in Chapter 5, distinguishing reference template extraction and extraction-based brute-force attacks applicable to BIOMFETRICS and countering them by limiting the number of identification requests of this system with an additional security margin. We then identified a weakness of FHE input leakage due to its single-party natures, and subsequently proposed a novel distributed decryption protocol named COLMADE to address it.

Chapter 7

Conclusion

The use of biometrics, such as fingerprints, facial recognition, or iris scans, has become popular in applications such as access control and financial transactions, but it also poses severe privacy concerns. Biometric data constitutes highly sensitive information that can be used to identify and track individuals, and it can be severely misused if it were to fall into the wrong hands. The main goal of this thesis is to protect the privacy of biometric data while still allowing its use for identification and authentication.

7.1 Contributions

This thesis is devoted to the study of privacy-preserving biometric systems, tackling multiple scenarios with varying security requirements and proposing innovative secure solutions suitable for real-world applications.

We kick-off this thesis providing an overview of generic biometric systems in Chapter 2, covering their components (reference and live templates, feature extractor based on CNNs, matching) and phases (enrollment and verification). We formulate the accuracy metrics used to evaluate biometric systems, while putting a particular focus on face identification. We then extract a set of requirements that privacy-preserving biometric systems should uphold: *privacy* of all the templates and the feature extractor (R1), *irreversibility* of the templates to avoid unintended reconstruction (R2), unlinkability of the templates to hide the identity of their owner (R3), correctness in the secure operations when dealing with malicious adversaries (R4), and accuracy & performance preservation in order to retain the characteristics of the original system (R5)-(R6). We then identified the main challenges that need to be addressed when building these systems, including dealing with *accuracy losses* (CH1) derived from the use of privacy-preserving techniques, balancing privacy with accuracy and performance (CH2), tackling *malicious adversaries* (CH3) to rely on weaker assumptions and obtain more robust systems, addressing *input leakage* in the output (CH4), and *implementing* ready-to-use solutions (CH5).

To fulfill these requirements and overcome the challenges we introduce the main privacy-preserving cryptographic tools at our disposal in Chapter 3, covering several variants of Multi-party Computation techniques (SS, RSS, ΠSS and FSS), two Fully Homomorphic Encryption schemes supporting SIMD (BFV and CKKS), and a Functional Encryption

Table 7.1: Requirements addressed at each chapter/section.

Chapter	**In biometric systems**	**Privacy - (R1)**	**Irreversibility - (R2)**	**Unlinkability - (R3)**	**Correctness - (R4)**	**Accuracy - (R5)**	**Performance - (R6)**	**Implementation - (CH5)**	**Title**	**Section**
3	-							●	Pyfhel	3.6
4	Feature extraction	●			●	●	●		Banners	4.1
5	Verification	●	◐	●		●	●		BiomFEtrics	5.2
		●	◐	●			●	●	Funshade	5.3
		●	◐	●		●	●		Grote	5.4
6	Output reveal		◑						Input leakage	6.1
		●	●		●			●	Colmade	6.2

scheme suitable for biometrics (FHIPE). We continue with a discussion on their relative strengths and weaknesses centered around the protection of the principal steps in biometric systems: the feature extraction, the template verification, and the output reveal. We comment on available implementations and their shortcomings, highlighting an inherent limitation in FHE libraries. Section 3.6 presents Pyfhel, a novel open-source[1] Python library designed to address these issues (CH5) and facilitate the development of FHE-based solutions to non-experts.

Chapter 4 addresses the protection of the feature extractor, presenting an original approach to protect Binarized Neural Networks by relying on Replicated Secret Sharing named Banners. We leverage RSS to provide privacy for the input data and the model (R1) and correctness guarantees (R4) against one malicious corruption in a three-party setting (CH3). We demonstrate how our protocol preserves the accuracy of BNNs while being on a par with the performance of equivalent state-of-the-art techniques.

Chapter 5 tackles the protection of biometric verification with the privacy-preserving tools at our disposal. We display three novel methods to protect the privacy of the biometric templates (R1), guaranteeing their unlinkability based on the computational hiding properties (R3), and making the protected templates (via encrypting or sharing) irreversible without access to the secret keys or all the computing parties (R2). These methods are:

a. BiomFEtrics (Section 5.2), a novel FHIPE solution optimized for online latency that allows for the private computation of scalar products between templates. We tune its latency (R6)-(CH2) to be of practical use in a one-time face identification scenario, preserving the accuracy of the original system (R5).

[1]Available in `https://github.com/ibarrond/Pyfhel`

b. FUNSHADE (Section 5.3), a new 2PC protocol based on a combination of ΠSS and FSS to securely compute various distance metrics followed by a comparison to a threshold against semi-honest adversaries. We evaluate its sub-millisecond latency (R6) to perform 1 : 1 biometric authentication with high degree of numerical precision, and we provide an open-source[2] ready-to-use implementation (CH5).

c. GROTE (Section 5.4), an innovative FHE-based algorithm for privacy-preserving face identification that reduces the number of costly non-linear operations in the encrypted domain based on group testing, later applied to CKKS-based face identification. We test this solution and ensure it maintains a high degree of biometric accuracy (R5), while improving the performance of the original system (R6) by a factor of 30% for reference databases of at least 8,000 identities.

Throughout this chapter, we also deal with various sources of accuracy losses (CH1), from the fixed-point encoding in BIOMFETRICS to the approximation errors in CKKS of GROTE.

Lastly, Chapter 6 tackles the problem of input leakage in the output of the biometric system (CH4). We first provide a formal definition of input leakage and its modeling using information theory in Section 6.1. As this leakage can compromise the irreversibility of protected templates (R2), we analyze the input leakage present in the three solutions proposed in Chapter 5, distinguishing reference template extraction and extraction-based brute-force attacks applicable to BIOMFETRICS and countering them by limiting the number of identification requests to this system with a carefully chosen security margin. We then identify a weakness of FHE input leakage due to its single-party nature, and subsequently propose a novel distributed decryption protocol named COLMADE to address it. COLMADE (Section 6.2) undertakes the minimization of input leakage in the output of privacy-preserving biometric systems (R1) with a novel solution that distributes the decryption of the ciphertexts among multiple parties and adds a distributed mask to ensure that only one bit of information is revealed in the output. This minimal output guarantees a high degree of irreversibility in the templates (R2). We provide an extension to ensure correctness of the decryption (R4) and provide an open-source[3] implementation of this protocol (CH5).

All in all, this thesis constitutes a comprehensive study of privacy-preserving biometric systems, and it presents a set of novel contributions that are ready to use as stepping stones to build secure and practical biometric solutions in a wide variety of scenarios.

7.2 Future Work

In Chapters 3 to 6 we propose multiple protocols to protect different aspects of biometric systems. As in any proposal, these solutions can be improved and built upon. In line with these works, future research is envisioned for the following topics:

- PYFHEL is to be enhanced to extend the set of supported libraries (e.g., OpenFHE [11], Helib [121]) while continuing our work of creating easy-to-use high-level APIs for low-level features. This project has gained traction in the community, driving several

[2]Available in `https://github.com/ibarrond/funshade`
[3]Available in `https://github.com/ibarrond/colmade`

new features such as partial support for distributed keys in both BFV (DBFV) and CKKS while also opening the door to new applications such as secure data analytics of sensitive data from the public sector (e.g., health data, tax information). Furthermore, we envision extending PYFHEL to support other homomorphic encryption schemes such as TFHE [65].

- Future steps of BANNERS could aim at Bit Slicing techniques to obtain considerable parallelization by leveraging on SIMD operations. Additionally, we envision the use of models trained specifically for biometric identification (e.g., face recognition). Although BANNERS is still slow for real time face recognition as-is, it can already be used for non time-constrained biometric applications, as well as to protect a subset of the layers in the biometric feature extractor. Lastly, we consider extending BANNERS to secure BNN training. In a similar line of work, we visualize the use of other non-binarized MPC-based solutions [149] for the protection of the CNNs that make up the feature extractor.

- The protections of biometric verification proposed in this thesis have so far focused on semi-honest adversaries. Future work could extend these protections to malicious adversaries. We contemplate enhancing FUNSHADE with maliciously secure techniques from SPDZ2k [77] for the linear function evaluations and the extensions to FSS suggested in [38] for the threshold comparison. Similarly, we consider upgrading GROTE with Verifiable FHE [237] to ensure the correctness of all the encrypted operations.

- The evaluation of all these solutions was carried out with face data from the LFW dataset [127]. We consider extending our experiments to substantially bigger datasets holding tens of thousands of identities, in an effort to scale up the biometric verification techniques for more demanding contexts. Likewise, these solutions can be extended to other biometric modalities such as fingerprint or iris, and even cover multimodal biometric systems.

- COLMADE is designed specifically for the BFV scheme. However, other schemes such as CKKS are more suitable for biometric applications thanks to the support of variable-scale encodings (and it is indeed the reason why we chose CKKS for GROTE). We envision extending COLMADE to CKKS, and possibly other schemes such as TFHE

- Lastly, we have dealt with each of the blocks in biometric systems separately, from the feature extraction (BANNERS) to the verification (BIOMFETRICS, FUNSHADE and GROTE) and the controlled output reveal (COLMADE). Designing an end-to-end solution for biometric systems that incorporates all these techniques and addresses its combined intricacies is an open problem and a promising future research direction.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC CCS Conference*, pages 308–318, 2016.

[3] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In *IACR International Workshop on Public Key Cryptography*, pages 128–157. Springer, 2019.

[4] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *IACR International Workshop on PKC*, pages 733–751. Springer, 2015.

[5] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *CRYPTO*, pages 597–627. Springer, 2018.

[6] Aysajan Abidin. On privacy-preserving biometric authentication. In *International Conference on Information Security and Cryptology*, pages 169–186. Springer, 2017.

[7] Insaf Adjabi, Abdeldjalil Ouahabi, Amir Benzaoui, and Abdelmalik Taleb-Ahmed. Past, present, and future of face recognition: A review. *Electronics*, 9(8):1188, 2020.

[8] Anshul Aggarwal, Trevor E Carlson, Reza Shokri, and Shruti Tople. Soteria: In search of efficient neural networks for private inference. *arXiv preprint arXiv:2007.12934*, 2020.

[9] Shashank Agrawal and Melissa Chase. Fame: fast attribute-based message encryption. In *Proceedings of the 2017 ACM SIGSAC CCS Conference*, pages 665–682, 2017.

[10] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Annual International Cryptology Conference*, pages 333–362. Springer, 2016.

[11] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, et al. Openfhe: Open-source fully homomorphic encryption library. In *Proceedings of*

the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pages 53–63, 2022.

[12] Milad Alizadeh, Javier Fernández-Marqués, Nicholas D. Lane, and Yarin Gal. A systematic study of binary neural networks' optimisation. In *International Conference on Learning Representations*, 2019.

[13] Xiang An, Jiankang Deng, Jia Guo, Ziyong Feng, XuHan Zhu, Jing Yang, and Tongliang Liu. Killing two birds with one stone: Efficient and robust training of face recognition cnns by partial fc. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4042–4051, 2022.

[14] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *Proceedings of the 2016 ACM SIGSAC CCS Conference*, pages 805–817, 2016.

[15] Nuttapong Attrapadung, Benoît Libert, and Elie De Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *International workshop on public key cryptography*, pages 90–108. Springer, 2011.

[16] Multiple authors. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[17] Jean-Claude Bajard, Julien Eynard, M Anwar Hasan, and Vincent Zucca. A full rns variant of fv like somewhat homomorphic encryption schemes. In *International Conference on Selected Areas in Cryptography*, pages 423–442. Springer, 2016.

[18] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *Annual International Cryptology Conference*, pages 67–98. Springer, 2017.

[19] Manuel Barbosa, Dario Catalano, Azam Soleimanian, and Bogdan Warinschi. Efficient function-hiding functional encryption: From inner-products to orthogonality. In *Cryptographers' Track at the RSA Conference*, pages 127–148. Springer, 2019.

[20] Mauro Barni, Tiziano Bianchi, Dario Catalano, Mario Di Raimondo, Ruggero Donida Labati, Pierluigi Failla, Dario Fiore, Riccardo Lazzeretti, Vincenzo Piuri, Alessandro Piva, et al. A privacy-compliant fingerprint recognition system based on homomorphic encryption and fingercode templates. In *2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–7. IEEE, 2010.

[21] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In *International Conference on Security and Cryptography for Networks*, pages 175–196. Springer, 2014.

[22] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Annual International Cryptology Conference*, pages 420–432. Springer, 1991.

[23] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31 –39, 2011.

[24] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption, 2021.

[25] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE SP symposium (SP'07)*, pages 321–334. IEEE, 2007.

[26] Joseph Bethge, Haojin Yang, Marvin Bornstein, and Christoph Meinel. Back to simplicity: How to train accurate bnns from scratch?, 2019.

[27] Marcelo Blatt, Alexander Gusev, Yuriy Polyakov, and Shafi Goldwasser. Secure large-scale genome-wide association studies using homomorphic encryption. *Proceedings of the National Academy of Sciences of the United States of America*, 117(21):11608–11613, 26 May 2020.

[28] Vishnu Naresh Boddeti. Secure face matching using fully homomorphic encryption. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–10. IEEE, 2018.

[29] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. Mp2ml: A mixed-protocol machine learning framework for private inference. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–10. Association for Computing Machinery, 2020.

[30] 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*, pages 528–558. Springer, 2021.

[31] Ruud M Bolle, Jonathan H Connell, Sharath Pankanti, Nalini K Ratha, and Andrew W Senior. *Guide to biometrics*. Springer Science & Business Media, 2013.

[32] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer, 2018.

[33] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[34] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.

[35] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. On bitcoin as a public randomness source. *Cryptology ePrint Archive*, 2015.

[36] Florian Bourse. *Functional encryption for inner-product evaluations*. PhD thesis, PSL Research University, 2017.

[37] Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In *Annual International Cryptology Conference*, pages 483–512. Springer, 2018.

[38] Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. Function secret sharing for mixed-mode and fixed-point secure computation. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part II*, pages 871–900, 2021.

[39] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT*, pages 337–367. Springer, 2015.

[40] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1292–1303, 2016.

[41] Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure computation with preprocessing via function secret sharing. In *17th International Conference on Theory of Cryptography, TCC 2019*, pages 341–371. Springer, 2019.

[42] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology – CRYPTO 2012*, pages 868–886. Springer Berlin Heidelberg, 2012.

[43] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science*, ITCS '12, pages 309–325, New York, NY, USA, 8 January 2012. ACM.

[44] Julien Bringer, Herve Chabanne, Melanie Favre, Alain Patey, Thomas Schneider, and Michael Zohner. Gshade: Faster privacy-preserving distance computation and biometric identification. In *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, pages 187–198, 2014.

[45] Julien Bringer, Hervé Chabanne, and Alain Patey. Shade: Secure hamming distance computation from oblivious transfer. In *International Conference on Financial Cryptography and Data Security*, pages 164–176. Springer, 2013.

[46] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[47] Adrian Bulat and Georgios Tzimiropoulos. XNOR-Net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019.

[48] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of CRYPTOLOGY*, 13(1):143–202, 2000.

[49] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[50] Ran Canetti, Ivan Damgaard, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 262–279. Springer, 2001.

[51] Sergiu Carpov, Nicolas Gama, Mariya Georgieva, and Juan Ramon Troncoso-Pastoriza. Privacy-preserving semi-parallel logistic regression training with fully homomorphic encryption. *BMC medical genomics*, 13(Suppl 7):88, 21 July 2020.

[52] Centers for Medicare & Medicaid. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Online at http://www.cms.hhs.gov/hipaa/, 1996.

[53] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. Privacy-preserving classification on deep neural network. *IACR Cryptol. ePrint Arch.*, 2017:35, 2017.

[54] Mahawaga Arachchige Pathum Chamikara, Peter Bertok, Ibrahim Khalil, Dongxi Liu, and Seyit Camtepe. Privacy preserving face recognition utilizing differential privacy. *Computers & Security*, 97:101951, 2020.

[55] Nishanth Chandran, Divya Gupta, Aseem Rastogi, Rahul Sharma, and Shardul Tripathi. Ezpc: programmable and efficient secure two-party computation for machine learning. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 496–511. IEEE, 2019.

[56] Konstantinos Chatzikokolakis, Tom Chothia, and Apratim Guha. Statistical measurement of information leakage. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 390–404. Springer, 2010.

[57] Harsh Chaudhari, Ashish Choudhury, Arpita Patra, and Ajith Suresh. Astra: high throughput 3pc over rings with application to secure prediction. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 81–92, 2019.

[58] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19, 1988.

[59] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 15 January 2018. ACM.

[60] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1243–1255, New York, NY, USA, 30 October 2017. Association for Computing Machinery.

[61] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full RNS variant of approximate homomorphic encryption. In *Selected Areas in Cryptography – SAC 2018*, pages 347–368. Springer International Publishing, 2019.

[62] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437. Springer International Publishing, 2017.

[63] Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 221–256. Springer, 2020.

[64] Jung Hee Cheon, Dongwoo Kim, Duhyeong Kim, Hun Hee Lee, and Keewoo Lee. Numerical method for comparison on homomorphically encrypted numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 415–445. Springer, 2019.

[65] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology. The Journal of the International Association for Cryptologic Research*, 33(1):34–91, 1 January 2020.

[66] Ilaria Chillotti, Marc Joye, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. CONCRETE: Concrete operates on ciphertexts rapidly by extending TfhE. In *WAHC 2020 – 8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 15 December 2020.

[67] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. Cryptology ePrint Archive, Report 2021/091, 2021.

[68] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. Cryptology ePrint Archive, Report 2021/091, 2021.

[69] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In *ASIACRYPT*, pages 703–732. Springer, 2018.

[70] Edward J Chou, Arun Gururajan, Kim Laine, Nitin Kumar Goel, Anna Bertiger, and Jack W Stokes. Privacy-preserving phishing web page classification via fully homomorphic encryption. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2792–2796. IEEE, 2020.

[71] Bismita Choudhury, Patrick Then, Biju Issac, Valliappan Raman, and Manas Kumar Haldar. A survey on biometrics and cancelable biometrics systems. *International Journal of Image and Graphics*, 18(01):1850006, 2018.

[72] Jeremy Clark and Urs Hengartner. On the use of financial data as a random beacon. In *2010 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 10)*, 2010.

[73] Ran Cohen. Secure multiparty computation: Introduction. Open web. `https://www.cs.tau.ac.il/~iftachh/Courses/Seminars/MPC/Intro.pdf`.

[74] Charles J Colbourn. Group testing for consecutive positives. *Annals of Combinatorics*, 3(1):37–41, 1999.

[75] European Commission. 2018 reform of eu data protection rules.

[76] European Commission. Proposal for a regulation laying down rules to prevent and combat child sexual abuse, May 2022.

[77] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. Spdz2k: Efficient mpc mod 2ˆ k for dishonest majority. *Cryptology ePrint Archive*, 2018.

[78] Anders Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. Technical report, Cryptology ePrint Archive, Report 2020/1330, 2020. https://eprint. iacr. org ..., 2020.

[79] Anders Dalskov, Daniel Escudero, and Marcel Keller. Secure evaluation of quantized neural networks. *Proceedings on Privacy Enhancing Technologies*, 2020(4):355–375, 2020.

[80] Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The tinytable protocol for 2-party secure computation, or: gate-scrambling revisited. In *Annual International Cryptology Conference*, pages 167–187. Springer, 2017.

[81] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.

[82] Shaveta Dargan and Munish Kumar. A comprehensive survey on the biometric recognition systems based on physiological and behavioral modalities. *Expert Systems with Applications*, 143:113114, 2020.

[83] Roshan Dathathri, Blagovesta Kostova, Olli Saarikivi, Wei Dai, Kim Laine, and Madanlal Musuvathi. EVA: An encrypted vector arithmetic language and compiler for efficient homomorphic computation. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 27 December 2019.

[84] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In *PKC 2016*, pages 164–195. Springer, 2016.

[85] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In *IACR International Workshop on Public Key Cryptography*, pages 245–277. Springer, 2018.

[86] John Daugman. How iris recognition works. In *The essential guide to image processing*, pages 715–739. Elsevier, 2009.

[87] Emiliano De Cristofaro, Kaitai Liang, and Yuruo Zhang. Privacy-preserving genetic relatedness test. *arXiv preprint arXiv:1611.03006*, 2016.

[88] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[89] Jiankang Deng, Jia Guo, Xue Niannan, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019.

[90] Jiankang Deng, Jia Guo, Zhou Yuxiang, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. In *arxiv*, 2019.

[91] Mark Driver. Emerging technologies: Homomorphic encryption for data sharing with privacy. Technical report, Gartner, Inc, 23 April 2020.

[92] Pawel Drozdowski, Nicolas Buchmann, Christian Rathgeb, Marian Margraf, and Christoph Busch. On the application of homomorphic encryption to face identification. In *2019 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–5. IEEE, 2019.

[93] Dingzhu Du, Frank K Hwang, and Frank Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 2000.

[94] Ling Du, Anthony TS Ho, and Runmin Cong. Perceptual hashing for image authentication: A survey. *Signal Processing: Image Communication*, 81:115713, 2020.

[95] Keita Emura. On the security of keyed-homomorphic pke: Preventing key recovery attacks and ciphertext validity attacks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 104(1):310–314, 2021.

[96] EPFL-LDS. Lattigo v2.4.0. Online: `https://github.com/ldsec/lattigo`, January 2022.

[97] Saroja Erabelli. *pyFHE-a Python library for fully homomorphic encryption.* PhD thesis, Massachusetts Institute of Technology, 2020.

[98] David Evans, Yan Huang, Jonathan Katz, and Lior Malka. Efficient privacy-preserving biometric identification. In *Proceedings of the 17th conference Network and Distributed System Security Symposium, NDSS*, volume 68, pages 90–98, 2011.

[99] Diana-Elena Fălămaş, Kinga Marton, and Alin Suciu. Assessment of two privacy preserving authentication methods using secure multiparty computation based on secret sharing. *Symmetry*, 13(5):894, 2021.

[100] J Fan and F Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012.

[101] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

[102] Justin Johnson Fei-Fei Li, Andrej Karpathy. CNNs in practice. Open Web slides from lecture, 2016. `http://cs231n.stanford.edu/slides/2016/winter1516_lecture11.pdf`.

[103] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[104] Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 225–255. Springer, 2017.

[105] Gartner. Gartner predictions on biometric authentication. *https://www.gartner.com/en/newsroom/press-releases/2019-02-05-gartner-predicts-increased-adoption-of-mobile-centric*, 2019.

[106] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *Proc. Priv. Enhancing Technol.*, 2017(4):345–364, 2017.

[107] Lukas Geiger and Plumerai Team. Larq: An open-source library for training binarized neural networks. *Journal of Open Source Software*, 5(45):1746, January 2020.

[108] California Attorney General. California consumer privacy act (ccpa), 2018.

[109] Craig Gentry et al. *A fully homomorphic encryption scheme*, volume 20. Stanford, 2009.

[110] 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*, pages 201–210, 2016.

[111] Babak Poorebrahim Gilkalaye, Ajita Rattani, and Reza Derakhshani. Euclidean-distance based fuzzy commitment scheme for biometric template security. In *2019 7th International Workshop on Biometrics and Forensics (IWBF)*, pages 1–6. IEEE, 2019.

[112] O Goldreich, S Micali, and A Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229, 1987.

[113] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications.* Cambridge university press, 2009.

[114] 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*, pages 536–553. Springer, 2013.

[115] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. The knowledge complexity of interactive proof-systems. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 203–225. ACM, 2019.

[116] Christian Gollier and Olivier Gossner. Group testing against covid-19. Technical report, EconPol Policy Brief, 2020.

[117] Laurent Gomez, Alberto Ibarrondo, Marcus Wilhelm, José Márquez, and Patrick Duverger. Security for distributed machine learning based software. In *International Conference on E-Business and Telecommunications*, pages 111–134, 2018.

[118] Marta Gomez-Barrero, Emanuele Maiorana, Javier Galbally, Patrizio Campisi, and Julian Fierrez. Multi-biometric template protection based on homomorphic encryption. *Pattern Recognition*, 67:149–163, 2017.

[119] 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*, pages 89–98, 2006.

[120] Patrick Grother, Mei Ngan, and Kayee Hanaoka. Ongoing face recognition vendor test (frvt) part 1: Verification. *National Institute of Standards and Technology*, 2022.

[121] Shai Halevi and Victor Shoup. Design and implementation of HElib: a homomorphic encryption library. Cryptology ePrint Archive, Report 2020/1481, 2020.

[122] Jutta Hämmerle-Uhl, Georg Penn, Gerhard Pötzelsberger, and Andreas Uhl. Size-reduction strategies for iris codes. *International Journal of Computer and Information Engineering*, 9(1):290–293, 2015.

[123] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[124] Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7533–7544. Curran Associates, Inc., 2019.

[125] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189*, 2017.

[126] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer, 2015.

[127] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[128] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[129] Huelse. SEAL-Python, 2020.

[130] Alberto Ibarrondo, Hervé Chabanne, Vincent Despiegel, and Melek Önen. Colmade: Collaborative masking in auditable decryption for bfv-based homomorphic encryption. In *Proceedings of the 2022 ACM Workshop on Information Hiding and Multimedia Security*, pages 129–139, 2022.

[131] Alberto Ibarrondo, Hervé Chabanne, Vincent Despiegel, and Melek Önen. Grote: Group testing for privacy-preserving face identification. In *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy*, 2023.

[132] Alberto Ibarrondo, Hervé Chabanne, and Melek Önen. Banners: Binarized neural networks with replicated secret sharing. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pages 63–74, 2021.

[133] Alberto Ibarrondo, Hervé Chabanne, and Melek Önen. Practical privacy-preserving face identification based on function-hiding functional encryption. In *International Conference on Cryptology and Network Security*, pages 63–71. Springer, 2021.

[134] Alberto Ibarrondo, Hervé Chabanne, and Melek Önen. Funshade: Functional secret sharing for two-party secure thresholded distance evaluation. *Cryptology ePrint Archive*, 2022.

[135] Alberto Ibarrondo and Melek Önen. Fhe-compatible batch normalization for privacy preserving deep learning. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 389–404. Springer, 2018.

[136] Alberto Ibarrondo and Alexander Viand. Pyfhel: Python for homomorphic encryption libraries. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 11–16, 2021.

[137] IDEMIA. Top 4 trends in biometrics for 2020. *https://www.idemia.com/news/idemias-top-4-trends-biometrics-2020-2020-01-28*, 2020.

[138] Ilia Iliashenko and Vincent Zucca. Faster homomorphic comparison operations for bgv and bfv. *Proceedings on Privacy Enhancing Technologies*, 2021(3):246–264, 2021.

[139] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[140] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *Theory of Cryptography Conference*, pages 600–620. Springer, 2013.

[141] Anil K Jain, Patrick Flynn, and Arun A Ross. *Handbook of biometrics.* Springer Science & Business Media, 2007.

[142] Seong-Yun Jeon and Mun-Kyu Lee. Acceleration of inner-pairing product operation for secure biometric verification. *Sensors*, 21(8):2859, 2021.

[143] Lei Jiang and Lei Ju. Fhebench: Benchmarking fully homomorphic encryption schemes. *arXiv preprint arXiv:2203.00728*, 2022.

[144] Zhe Jin, Jung Yeon Hwang, Yen-Lung Lai, Soohyung Kim, and Andrew Beng Jin Teoh. Ranking-based locality sensitive hashing-enabled cancelable biometrics: Index-of-max hashing. *IEEE Transactions on Information Forensics and Security*, 13(2):393–407, 2017.

[145] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019.

[146] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1651–1669, 2018.

[147] Sreekanth Kannepalli, Kim Laine, and Radames Cruz Moreno. Password monitor: Safeguarding passwords in microsoft edge, 21 January 2021. Accessed: 2021-7-5.

[148] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *annual international conference on the theory and applications of cryptographic techniques*, pages 146–162. Springer, 2008.

[149] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. Cryptology ePrint Archive, Report 2020/521, 2020. `https://eprint.iacr.org/2020/521`.

[150] Andrey Kim, Antonis Papadimitriou, and Yuriy Polyakov. Approximate homomorphic encryption with reduced approximation error. In *Cryptographers' Track at the RSA Conference*, pages 120–144. Springer, 2022.

[151] Andrey Kim, Yongsoo Song, Miran Kim, Keewoo Lee, and Jung Hee Cheon. Logistic regression model training based on the approximate homomorphic encryption. *BMC medical genomics*, 11(4):23–31, 2018.

[152] Miran Kim, Arif Harmanci, Jean-Philippe Bossuat, Sergiu Carpov, Jung Hee Cheon, Ilaria Chillotti, Wonhee Cho, David Froelicher, Nicolas Gama, Mariya Georgieva, Seungwan Hong, Jean-Pierre Hubaux, Duhyeong Kim, Kristin Lauter, Yiping Ma, Lucila Ohno-Machado, Heidi Sofia, Yongha Son, Yongsoo Song, Juan Troncoso-Pastoriza, and Xiaoqian Jiang. Ultra-fast homomorphic encryption models enable secure outsourcing of genotype imputation. *bioRxiv*, 2020.

[153] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J Wu. Function-hiding inner product encryption is practical. In *SCN18*, pages 544–562. Springer, 2018.

[154] Sungwook Kim, Jinsu Kim, and Jae Hong Seo. A new approach to practical function-private inner product encryption. *Theoretical Computer Science*, 783:22–40, 2019.

[155] Jascha Kolberg, Pia Bauspieß, Marta Gomez-Barrero, Christian Rathgeb, Markus Dürmuth, and Christoph Busch. Template protection based on homomorphic encryption: Computationally efficient application to iris-biometric verification and identification. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2019.

[156] Jascha Kolberg, Pawel Drozdowski, Marta Gomez-Barrero, Christian Rathgeb, and Christoph Busch. Efficiency analysis of post-quantum-secure face template protection schemes based on homomorphic encryption. In *2020 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–4. IEEE, 2020.

[157] Yassin Kortli, Maher Jridi, Ayman Al Falou, and Mohamed Atri. Face recognition systems: A survey. *Sensors*, 20(2):342, 2020.

[158] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 336–353. IEEE, 2020.

[159] Nitin Kumar et al. Cancelable biometrics: a comprehensive survey. *Artificial Intelligence Review*, 53(5):3403–3446, 2020.

[160] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. *SIAM Journal on Computing*, 39(5):2090–2112, 2010.

[161] LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[162] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[163] Eunsang Lee, Joon-Woo Lee, Young-Sik Kim, and Jong-Seon No. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Transactions on Dependable and Secure Computing*, 2021.

[164] Eunsang Lee, Joon-Woo Lee, Young-Sik Kim, and Jong-Seon No. Optimization of homomorphic comparison algorithm on rns-ckks scheme. *IEEE Access*, 10:26163–26176, 2022.

[165] Joohee Lee, Dongwoo Kim, Duhyeong Kim, Yongsoo Song, Junbum Shin, and Jung Hee Cheon. Instant privacy-preserving biometric authentication for hamming distance. *IACR Cryptol. ePrint Arch.*, 2018:1214, 2018.

[166] Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 10:30039–30054, 2022.

[167] Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. High-precision bootstrapping of rns-ckks homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 618–647. Springer, 2021.

[168] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In *Advances in Cryptology – EUROCRYPT 2021*, pages 648–677. Springer International, 2021.

[169] Baiqiang Liang, Hongrong Ding, Lianfang Huang, Haiqing Luo, and Xiao Zhu. Gwas in cancer: progress and challenges. *Molecular Genetics and Genomics*, pages 1–25, 2020.

[170] Damien Ligier, Sergiu Carpov, Caroline Fontaine, and Renaud Sirdey. Information leakage analysis of inner-product functional encryption based data classification. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 303–3035, 2017.

[171] Yehuda Lindell. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography*, pages 277–346, 2017.

[172] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–631, 2017.

[173] Chun Lo, Mingyan Liu, Jerome P Lynch, and Anna C Gilbert. Efficient sensor fault detection using combinatorial group testing. In *2013 IEEE international conference on distributed computing in sensor systems*, pages 199–206. IEEE, 2013.

[174] Jake Loftus, Alexander May, Nigel P Smart, and Frederik Vercauteren. On cca-secure somewhat homomorphic encryption. In *International Workshop on Selected Areas in Cryptography*, pages 55–72. Springer, 2011.

[175] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.

[176] Qian Lou and Lei Jiang. Hemet: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture. In *International Conference on Machine Learning*, pages 7102–7110. PMLR, 2021.

[177] Giulio Lovisotto, Raghav Malik, Ivo Sluganovic, Marc Roeschlin, Paul Trueman, and Ivan Martinovic. Mobile biometrics in financial services: A five factor framework. *University of Oxford, Oxford, UK*, 2017.

[178] Wen-jie Lu and Jun Sakuma. More practical privacy-preserving machine learning as a service via efficient secure matrix multiplication. In *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 25–36, 2018.

[179] Ying Luo, S Cheung Sen-ching, and Shuiming Ye. Anonymous biometric access control based on homomorphic encryption. In *2009 IEEE International Conference on Multimedia and Expo*, pages 1046–1049. IEEE, 2009.

[180] 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*, pages 1–23. Springer, 2010.

[181] T Soni Madhulatha. An overview on clustering methods. *arXiv preprint arXiv:1205.1117*, 2012.

[182] Suren Makaju, PWC Prasad, Abeer Alsadoon, AK Singh, and A Elchouemi. Lung cancer detection using ct scan images. *Procedia Computer Science*, 125:107–114, 2018.

[183] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[184] Yan Michalevsky and Marc Joye. Decentralized policy-hiding attribute-based encryption with receiver privacy. *IACR Cryptol. ePrint Arch.*, 2018:753, 2018.

[185] Payman Mohassel and Peter Rindal. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 35–52, 2018.

[186] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *Cryptology ePrint Archive*, 2020.

[187] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical cca2-secure and masked ring-lwe implementation. *Cryptology ePrint Archive*, 2016.

[188] Margarita Osadchy, Benny Pinkas, Ayman Jarrous, and Boaz Moskovich. Scifi-a system for secure face identification. In *2010 IEEE Symposium on Security and Privacy*, pages 239–254. IEEE, 2010.

[189] Dailé Osorio-Roig, Christian Rathgeb, Pawel Drozdowski, and Christoph Busch. Stable hash generation for efficient privacy-preserving face identification. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2021.

[190] Elena Pagnin and Aikaterini Mitrokotsa. Privacy-preserving biometric authentication: challenges and directions. *Security and Communication Networks*, 2017, 2017.

[191] Pascal Paillier. Public-Key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — EUROCRYPT '99*, pages 223–238. Springer Berlin Heidelberg, 1999.

[192] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[193] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. {ABY2. 0}: Improved {Mixed-Protocol} secure {Two-Party} computation. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2165–2182, 2021.

[194] Christine Payne. Musenet, 2019. *URL https://openai. com/blog/musenet*, 2019.

[195] Zhiniang Peng. Danger of using fully homomorphic encryption: A look at microsoft seal. *arXiv preprint arXiv:1906.07127*, 2019.

[196] Yuriy Polyakov, Kurt Rohloff, and Gerard W Ryan. PALISADE lattice cryptography library user manual. Technical report, NJIT, 2017.

[197] Fentec project. Cifer: Functional encryption library. `https://github.com/fentec-project/CiFEr`, 2021.

[198] P Punithavathi and Geetha Subbiah. Can cancellable biometrics preserve privacy? *Biometric Technology Today*, 2017(7):8–11, 2017.

[199] Michael O Rabin. How to exchange secrets with oblivious transfer. *Cryptology ePrint Archive*, 2005.

[200] Shantanu Rane, Ye Wang, Stark C Draper, and Prakash Ishwar. Secure biometrics: Concepts, authentication architectures, and challenges. *IEEE Signal Processing Magazine*, 30(5):51–64, 2013.

[201] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[202] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 325–342, 2020.

[203] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[204] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *In STOC*, pages 84–93. ACM Press, 2005.

[205] Oscar Reparaz, Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Additively homomorphic ring-lwe masking. In *Post-Quantum Cryptography*, pages 233–244. Springer, 2016.

[206] Oscar Reparaz, Sujoy Sinha Roy, Ruan De Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-lwe. *Journal of Cryptographic Engineering*, 6(2):139–153, 2016.

[207] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. XONN: XNOR-based oblivious deep neural network inference. In *Proceedings of the 28th USENIX Conference on Security Symposium*, page 1501–1518, USA, 2019. USENIX Association.

[208] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 707–721, 2018.

[209] R L Rivest, A Shamir, and L Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1 February 1978.

[210] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[211] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[212] Zhang Rui and Zheng Yan. A survey on biometric authentication: Toward secure and privacy-preserving identification. *IEEE access*, 7:5994–6009, 2018.

[213] Théo Ryffel, Pierre Tholoniat, David Pointcheval, and Francis Bach. Ariann: Low-interaction privacy-preserving deep learning via function secret sharing. *Proceedings on Privacy Enhancing Technologies*, 1:291–316, 2022.

[214] T Sabhanayagam, V Prasanna Venkatesan, and K Senthamaraikannan. A comprehensive survey on various biometric systems. *International Journal of Applied Engineering Research*, 13(5):2276–2297, 2018.

[215] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Efficient privacy-preserving face recognition. In *International conference on information security and cryptology*, pages 229–244. Springer, 2009.

[216] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473. Springer, 2005.

[217] Marios Savvides, BVK Vijaya Kumar, and Pradeep K Khosla. Cancelable biometric filters for face recognition. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 922–925. IEEE, 2004.

[218] Walter J Scheirer and Terrance E Boult. Cracking fuzzy vaults and biometric encryption. In *2007 Biometrics Symposium*, pages 1–6. IEEE, 2007.

[219] Microsoft SEAL (release 3.7). `https://github.com/Microsoft/SEAL`, September 2021. Microsoft Research, Redmond, WA.

[220] Adi Shamir. How to share a secret. *Comm. of the ACM*, 22(11):612–613, 1979.

[221] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[222] Koen Simoens, Julien Bringer, Hervé Chabanne, and Stefaan Seys. A framework for analyzing template security and privacy in biometric authentication systems. *IEEE Transactions on Information forensics and security*, 7(2):833–841, 2012.

[223] Taylor Simons and Dah-Jye Lee. A review of binarized neural networks. *Electronics*, 8(6), 2019.

[224] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[225] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.

[226] Stack Overflow. Stack overflow developer survey 2020, 2020. Accessed: 2021-7-5.

[227] Douglas R Stinson, Tran Van Trung, and Ruizhong Wei. Secure frameproof codes, key distribution patterns, group testing algorithms and related structures. *Journal of Statistical Planning and Inference*, 86(2):595–617, 2000.

[228] Yagiz Sutcu, Husrev Taha Sencar, and Nasir Memon. A secure biometric authentication scheme based on robust hashing. In *Proceedings of the 7th Workshop on Multimedia and Security*, pages 111–116, 2005.

[229] Veeru Talreja, Matthew C Valenti, and Nasser M Nasrabadi. Deep hashing for secure multimodal biometrics. *IEEE Transactions on Information Forensics and Security*, 16:1306–1321, 2020.

[230] Hiroto Tamiya, Toshiyuki Isshiki, Kengo Mori, Satoshi Obana, and Tetsushi Ohki. Improved post-quantum-secure face template protection system based on packed homomorphic encryption. In *2021 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–5. IEEE, 2021.

[231] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

[232] Mayada Tarek, Osama Ouda, and Taher Hamza. Robust cancellable biometrics scheme based on neural networks. *IET Biometrics*, 5(3):220–228, 2016.

[233] Alexander J. Titus, Shashwat Kishore, Todd Stavish, Stephanie M. Rogers, and Karl Ni. Pyseal: A python wrapper implementation of the seal homomorphic encryption library, 2018.

[234] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In *ICIS*, pages 408–425. Springer, 2016.

[235] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

[236] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pages 586–587. IEEE Computer Society, 1991.

[237] Alexander Viand, Christian Knabenhans, and Anwar Hithnawi. Verifiable fully homomorphic encryption. *arXiv preprint arXiv:2301.07041*, 2023.

[238] Paulo Vitorino, Sandra Avila, Mauricio Perez, and Anderson Rocha. Leveraging deep neural networks to fight child pornography in the age of social media. *Journal of Visual Communication and Image Representation*, 50:303–313, 2018.

[239] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: Efficient and private neural network training. *IACR Cryptol. ePrint Arch.*, 2018:442, 2018.

[240] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. FALCON: Honest-majority maliciously secure framework for private deep learning. *arXiv preprint arXiv:2004.02229*, 2020.

[241] Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. Splinter: Practical private queries on public data. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 299–313, 2017.

[242] Wikipedia. Ripple carry adder. Open web, 2020. `https://en.wikipedia.org/wiki/Adder_(electronics)#Ripple-carry_adder`.

[243] Peter Wolf, Abdul Alim, Brown Kasaro, Pontius Namugera, Mohammed Saneem, and Tamir Zorigt. *Introducing biometric technology in elections.* International Institute for Democracy and Electoral Assistance . . . , 2017.

[244] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.

[245] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Packed homomorphic encryption based on ideal lattices and its application to biometrics. In *International Conference on Availability, Reliability, and Security*, pages 55–74. Springer, 2013.

[246] Yang Yu, Zhiqiang Gong, Ping Zhong, and Jiaxin Shan. Unsupervised representation learning with deep convolutional neural network for remote sensing images. In *International Conference on Image and Graphics*, pages 97–108. Springer, 2017.

[247] Naser Zaeri. Minutiae-based fingerprint extraction and recognition. *Biometrics*, 2011.

[248] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[249] Kai Zhou and Jian Ren. Passbio: Privacy-preserving user-centric biometric authentication. *IEEE Transactions on Info. Forensics and Security*, 13(12):3050–3063, 2018.

[250] Youwen Zhu and Tsuyoshi Takagi. Efficient scalar product protocol and its privacy–preserving application. *International Journal of Electronic Security and Digital Forensics*, 7(1):1–19, 2015.

Appendix A

Appendix

A.1 Security Proofs of BANNERS Protocols

We apply the *ideal VS real world* simulation from [112] [48] [49] to prove the security of our protocols in Chapter 4, instantiating it for the setup of BANNERS. As mentioned in Section 3.5, we consider a static corruption model where the adversary must choose which participant to corrupt before the execution of the computations.

Our protocols preserve security in all contexts that use them as a black box, providing inputs and fresh randomness and using only the outputs. As such, they achieve universal composability. To prove that our protocols are secure under general composition, we rely on the Theorem 1.2 of [160].

The ideal world simulation contains an additional trusted party that receives all the inputs from all the standard parties, computes the ideal functionality correctly and sends the corresponding results to the standard parties. Conversely, the real world simulation executes the protocol as described in the BANNERS algorithms in the presence of adversaries (one malicious adversary in our case). We then prove our protocols secure by verifying that for every adversary in the real interaction, there exists a simulator in the ideal interaction such that the environment cannot distinguish between the two scenarios. In other words, whatever information the adversary extracts in the real interaction, the simulator can extract it in the ideal world as well. Thus, the entire blackbox behavior and interactions (inputs, outputs, communication) of all the parties in the real world scenario is statistically equivalent to that of the ideal world with the trusted party for the given adversary.

To prove each algorithm being secure, we replace it by their corresponding ideal functionality and then prove that the interactions can be simulated, then showing that the real and ideal interactions are indistinguishable from each other.

We set party P_k to be corrupt. In the ideal world, the simulator interacts with the adversary P_k and simulates exact transcripts for interactions between P_k and the other two honest parties, P_i, P_j. In the real world, P_k interacts with P_i, P_j directly. In all these cases, the simulator is able to extract the inputs from P_k by using only the values for the inputs of honest P_i & P_j, since our 2-out-of-3 RSS scheme requires only two shares to reconstruct a secret. These inputs are fed to the ideal/real functionality to generate correct output distributions.

Ideal Functionality 1 Integer-Binary VDP:

Players: Functionality receives integer shares $\langle\langle x_j \rangle\rangle$, $\langle\langle y_j \rangle\rangle \in \mathbb{Z}_{2^l}$

Output: Compute the element-wise multiplication of $\langle\langle x_j \rangle\rangle * \langle\langle y_j \rangle\rangle$, the cumulative addition of $\langle\langle \Sigma_{VDP} \rangle\rangle$, and sends resulting shares back to parties.

Theorem 6. *The Integer-Binary VDP algorithm (algorithm 7) securely realizes the Integer-Binary VDP functionality with abort in the presence of one malicious party.*

Proof. The simulator $\mathcal{S}$ for a malicious adversary corrupting P_k plays the role of the trusted party. To be able to simulate, we need to show that:

- Real interaction transcripts can be simulated.
- Honest parties P_i, P_j receive their outputs correctly.

Simulation can be easily derived from the standard maliciously secure multiplication of [104] in the integer case, also treated in theorem 2 of [185], along with theorem 5 of [185] for the binary to arithmetic conversion. The simulator for this multiplication can simulate the transcripts from steps 2-7. Note that the distributions of all random values (such as the first 2 SS shares in a triplet) are all uniform and hence achieve perfect security. Local steps such as 1 and 8 do not need simulation. If the protocol aborts at any time in the internal run, then the simulator sends Abort to the functionalities. Otherwise, it inputs the extracted shares of P_k along with those of P_i, P_j, and the parties receive their outputs. $\square$

Ideal Functionality 2 Binary BN + BA:

Players: Functionality receives arithmetic shares $\langle\langle x \rangle\rangle$, $\langle\langle \beta/\gamma \rangle\rangle \in \mathbb{Z}_{2^l}$.

Output: Computes local subtraction $\langle\langle x \rangle\rangle - \langle\langle \beta/\gamma \rangle\rangle$, followed by reconstruction extraction of the MSB. Generates and sends binary shares of result.

Theorem 7. *The Binary BN+BA algorithm (algorithm 8) securely realizes the Binary BN+BA ideal functionality with abort in the presence of one malicious party.*

Proof. The simulation can be derived from the standard maliciously secure integer addition (step 1) from [104], along with the theorem 2 of [240] for the binary activation. Abort conditions and outputs of honest parties apply just like in the previous proof. $\square$

Ideal Functionality 3 Binary VDP:

Players: The functionality receives binary shares of $[\![x_j]\!]$ in a given window with N elements.

Output: Computes element-wise XOR, transforms the resulting binary shares into arithmetic (with two intermediate AND gates) and performs local cumulative addition. Generates and sends shares of Res_{VDP} to parties.

Theorem 8. *The Binary VDP algorithm (algorithm 9) securely realizes the Binary VDP ideal functionality with abort in the presence of one malicious party.*

Proof. We apply a similar logic to that of the first functionality. The simulation can be derived from the standard maliciously secure XOR of [104] in the binary case (step 1) as well as secure integer addition (step 8), along with theorem 5 of [185] for the binary to arithmetic conversion (steps 2-7). Local steps such as 1 and 8 do not need simulation. Abort conditions and outputs of honest parties apply just like in the previous proof. □

Ideal Functionality 4 Maxpool:

Players: Functionality receives N binary shares $[\![x_j]\!]$ over a window.

Output: Computes local NOT on all shares followed by cumulative AND (following [104]) and a final NOT. Generates and sends binary shares of $[\![m]\!]_{maxpool}$ to parties.

Theorem 9. *The Maxpool algorithm (algorithm 10) securely realizes the Maxpool ideal functionality with abort in the presence of one malicious party.*

Proof. We apply composability of maliciously secure AND of [104] in the binary case (steps 3-5). Local negation of steps 1 and 5 does not need simulation. □

A.2 Banners experiments with CIFAR-10 dataset

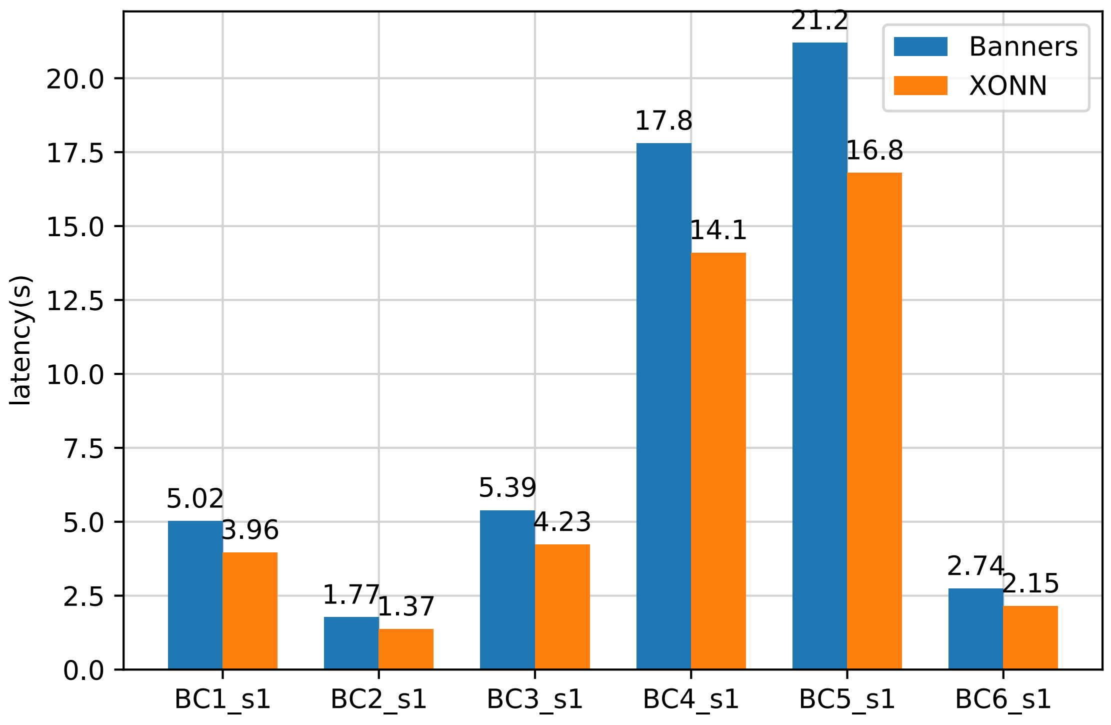


Figure A.1: Comparison in latency for CIFAR10 BNN models

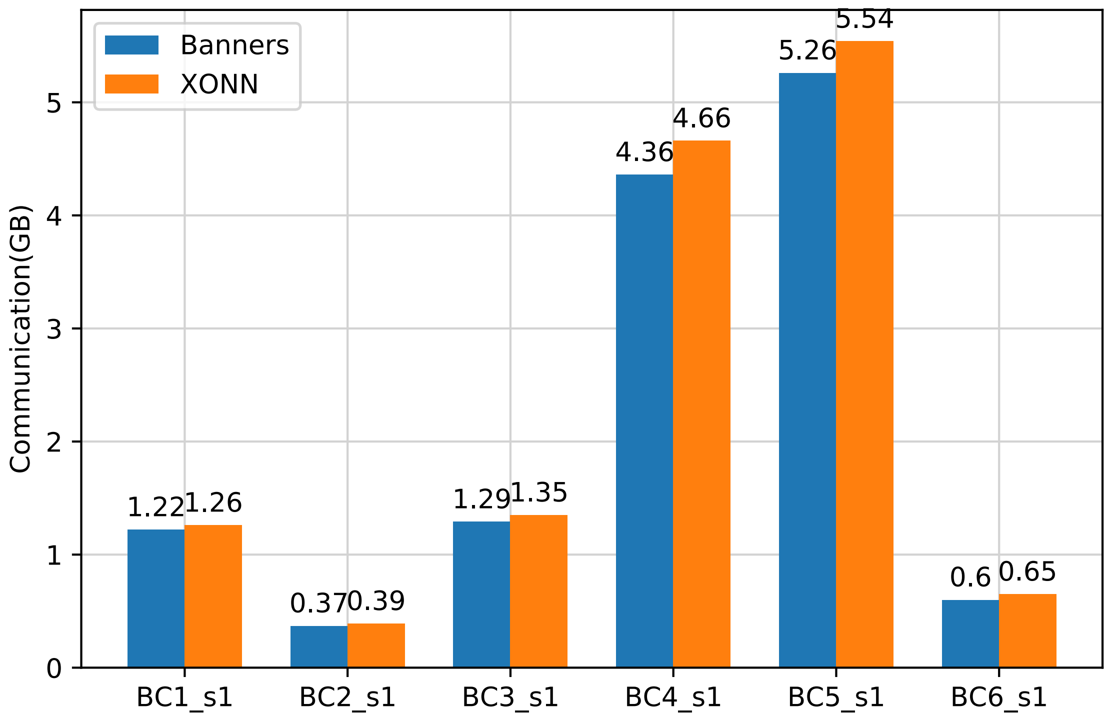


Figure A.2: Comparing communication for CIFAR10 BNNs

A.3 Side-by-side comparison of FunShade vs. AriaNN

A.4 Discussion on the realistic entropy of the input space

Beyond this, the attacker could also resort to prior knowledge of the template space (obtainable from feature extractors with similar characteristics) and project the partially extracted template to it, further increasing the chances of a successful impersonation.

The attacker could rely on prior latent space information obtainable from the non-protected feature extractor to narrow it down to a small subset of candidate templates that might be close enough to the reference template (in the cosine similarity sense) to serve as a fairly accurate and valid (allowing impersonation) approximations.

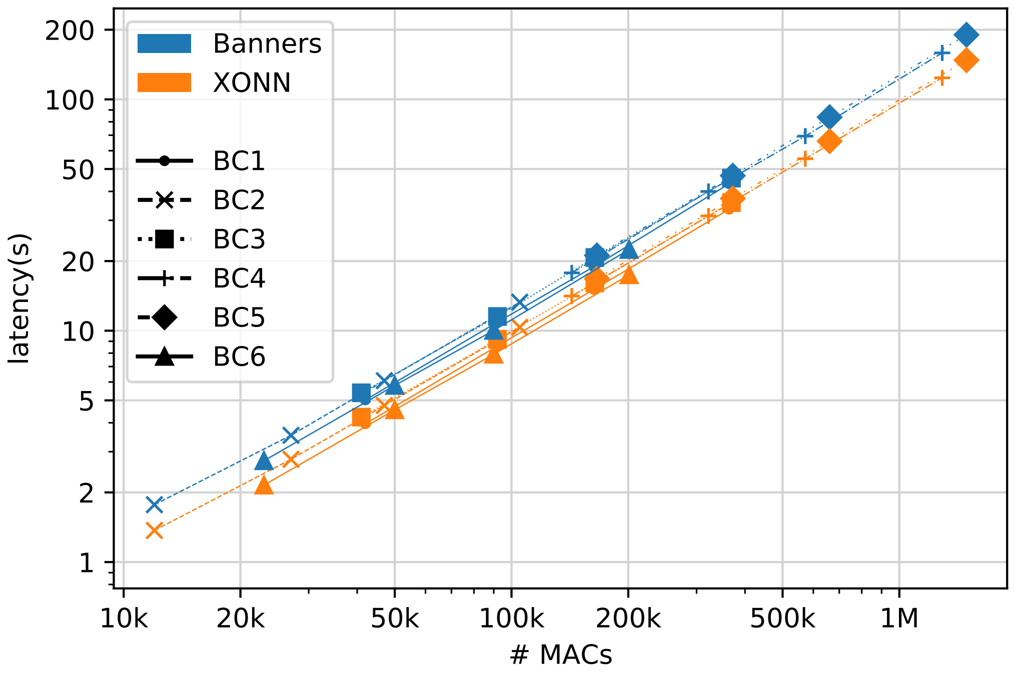


Figure A.3: Tradeoff between MACs and Latency for CIFAR10 BNN models

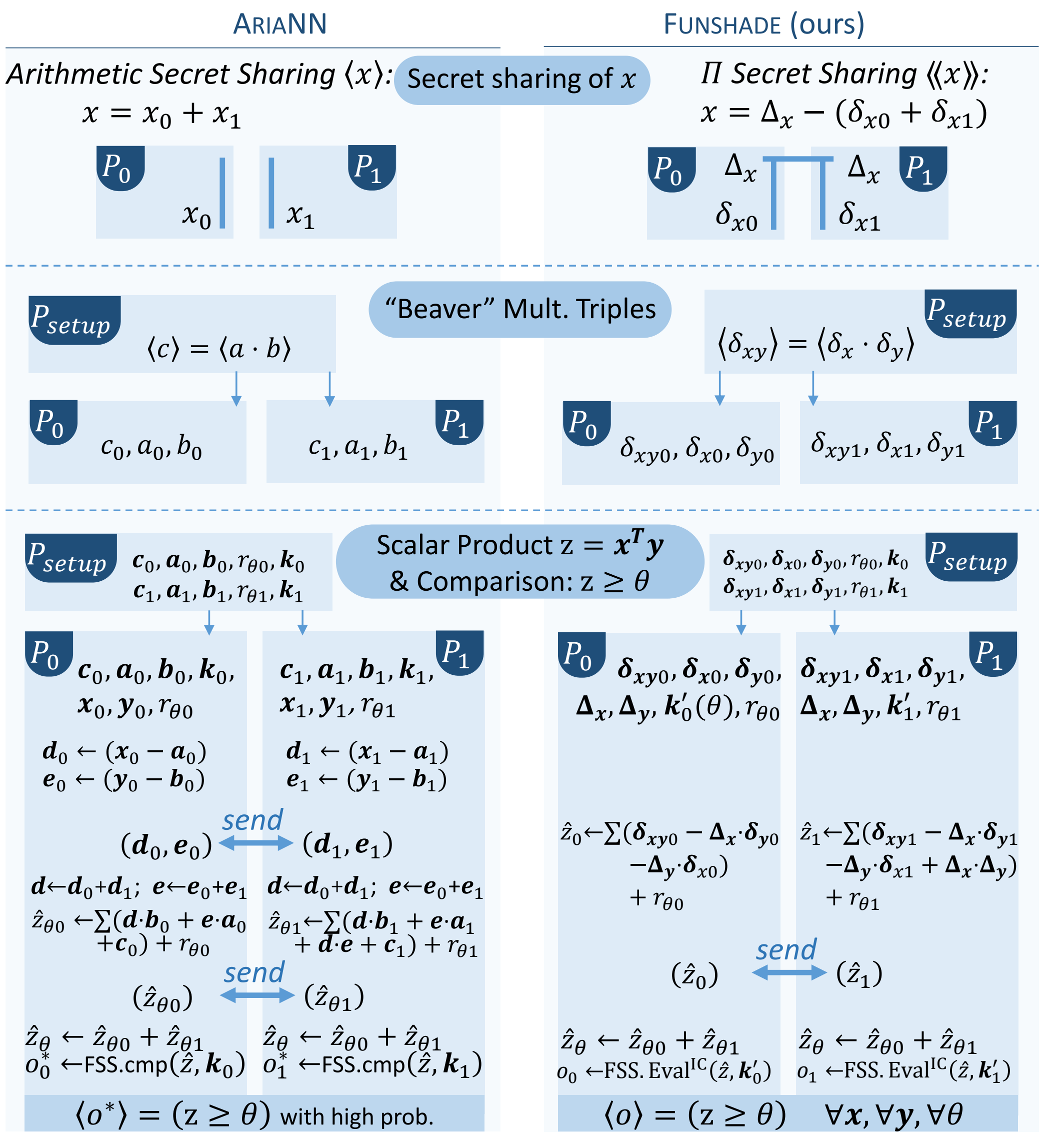

AriaNN
Funshade (ours)
Secret sharing of x
Arithmetic Secret Sharing $\langle x \rangle$: $x = x_0 + x_1$
P_0 x_0
P_1 x_1
Π Secret Sharing $\langle\langle x \rangle\rangle$: $x = \Delta_x - (\delta_{x0} + \delta_{x1})$
P_0 Δ_x δ_{x0}
P_1 Δ_x δ_{x1}
"Beaver" Mult. Triples
P_{setup} $\langle c \rangle = \langle a \cdot b \rangle$
P_0 c_0, a_0, b_0
P_1 c_1, a_1, b_1
P_{setup} $\langle \delta_{xy} \rangle = \langle \delta_x \cdot \delta_y \rangle$
P_0 $\delta_{xy0}, \delta_{x0}, \delta_{y0}$
P_1 $\delta_{xy1}, \delta_{x1}, \delta_{y1}$
Scalar Product $z = \boldsymbol{x}^T \boldsymbol{y}$ & Comparison: $z \geq \theta$
P_{setup} $\boldsymbol{c}_0, \boldsymbol{a}_0, \boldsymbol{b}_0, r_{\theta 0}, \boldsymbol{k}_0$ $\boldsymbol{c}_1, \boldsymbol{a}_1, \boldsymbol{b}_1, r_{\theta 1}, \boldsymbol{k}_1$
P_0 $\boldsymbol{c}_0, \boldsymbol{a}_0, \boldsymbol{b}_0, \boldsymbol{k}_0, \boldsymbol{x}_0, \boldsymbol{y}_0, r_{\theta 0}$
$\boldsymbol{d}_0 \leftarrow (\boldsymbol{x}_0 - \boldsymbol{a}_0)$ $\boldsymbol{e}_0 \leftarrow (\boldsymbol{y}_0 - \boldsymbol{b}_0)$
P_1 $\boldsymbol{c}_1, \boldsymbol{a}_1, \boldsymbol{b}_1, \boldsymbol{k}_1, \boldsymbol{x}_1, \boldsymbol{y}_1, r_{\theta 1}$
$\boldsymbol{d}_1 \leftarrow (\boldsymbol{x}_1 - \boldsymbol{a}_1)$ $\boldsymbol{e}_1 \leftarrow (\boldsymbol{y}_1 - \boldsymbol{b}_1)$
send
$(\boldsymbol{d}_0, \boldsymbol{e}_0)$ $(\boldsymbol{d}_1, \boldsymbol{e}_1)$
$\boldsymbol{d} \leftarrow \boldsymbol{d}_0 + \boldsymbol{d}_1$; $\boldsymbol{e} \leftarrow \boldsymbol{e}_0 + \boldsymbol{e}_1$
$\hat{z}_{\theta 0} \leftarrow \sum(\boldsymbol{d} \cdot \boldsymbol{b}_0 + \boldsymbol{e} \cdot \boldsymbol{a}_0 + \boldsymbol{c}_0) + r_{\theta 0}$
$\hat{z}_{\theta 1} \leftarrow \sum(\boldsymbol{d} \cdot \boldsymbol{b}_1 + \boldsymbol{e} \cdot \boldsymbol{a}_1 + \boldsymbol{d} \cdot \boldsymbol{e} + \boldsymbol{c}_1) + r_{\theta 1}$
send
$(\hat{z}_{\theta 0})$ $(\hat{z}_{\theta 1})$
$\hat{z}_\theta \leftarrow \hat{z}_{\theta 0} + \hat{z}_{\theta 1}$
$o_0^* \leftarrow$ FSS.cmp$(\hat{z}, \boldsymbol{k}_0)$
$o_1^* \leftarrow$ FSS.cmp$(\hat{z}, \boldsymbol{k}_1)$
$\langle o^* \rangle = (z \geq \theta)$ with high prob.
P_{setup} $\boldsymbol{\delta}_{xy0}, \boldsymbol{\delta}_{x0}, \boldsymbol{\delta}_{y0}, r_{\theta 0}, \boldsymbol{k}_0$ $\boldsymbol{\delta}_{xy1}, \boldsymbol{\delta}_{x1}, \boldsymbol{\delta}_{y1}, r_{\theta 1}, \boldsymbol{k}_1$
P_0 $\boldsymbol{\delta}_{xy0}, \boldsymbol{\delta}_{x0}, \boldsymbol{\delta}_{y0}, \boldsymbol{\Delta}_x, \boldsymbol{\Delta}_y, \boldsymbol{k}'_0(\theta), r_{\theta 0}$
P_1 $\boldsymbol{\delta}_{xy1}, \boldsymbol{\delta}_{x1}, \boldsymbol{\delta}_{y1}, \boldsymbol{\Delta}_x, \boldsymbol{\Delta}_y, \boldsymbol{k}'_1, r_{\theta 1}$
$\hat{z}_0 \leftarrow \sum(\boldsymbol{\delta}_{xy0} - \boldsymbol{\Delta}_x \cdot \boldsymbol{\delta}_{y0} - \boldsymbol{\Delta}_y \cdot \boldsymbol{\delta}_{x0}) + r_{\theta 0}$
$\hat{z}_1 \leftarrow \sum(\boldsymbol{\delta}_{xy1} - \boldsymbol{\Delta}_x \cdot \boldsymbol{\delta}_{y1} - \boldsymbol{\Delta}_y \cdot \boldsymbol{\delta}_{x1} + \boldsymbol{\Delta}_x \cdot \boldsymbol{\Delta}_y) + r_{\theta 1}$
send
$(\hat{z}_0)$ $(\hat{z}_1)$
$\hat{z}_\theta \leftarrow \hat{z}_{\theta 0} + \hat{z}_{\theta 1}$
$o_0 \leftarrow$ FSS.Eval$^{\mathrm{IC}}(\hat{z}, \boldsymbol{k}'_0)$
$o_1 \leftarrow$ FSS.Eval$^{\mathrm{IC}}(\hat{z}, \boldsymbol{k}'_1)$
$\langle o \rangle = (z \geq \theta)$ $\forall \boldsymbol{x}, \forall \boldsymbol{y}, \forall \theta$

Figure A.4: Side-by-side comparison between AriaNN and Funshade (ours)

List of Figures

List of Tables

List of Acronyms

ABE - Attribute-Based Encryption
AUC - Area Under the Curve
BA - Binary Activation
BFV - Brakerski / Fan-Vaikuntanathan
BGV - Brakerski-Gentry-Vaikuntanathan
BIP - Biometric Identity Provider
BN - Batch Normalization
BNN - Binarized Neural Network
BP - Biometric Provider
CCA - Chosen Ciphertext Attack
CKKS - Cheon-Kim-Kim-Song
CPA - Chosen Plaintext Attack
CNN - Convolutional Neural Network
CRT - Chinese Remainder Theorem
CSAM - Child Sexual Abuse Material
Conv - Convolution
CPA - Chosen Plaintext Attack
DB - Database
DBFV - Decentralized BFV
DL - Deep Learning
EER - Equal Error Rate
FAR - False Acceptance Rate
FC - Fully Connected
FE - Functional Encryption
FHIPE - Function-Hiding Inner Product Encryption
FHE - Fully Homomorphic Encryption
FRR - False Rejection Rate
FSS - Functional Secret Sharing
GB - GigaByte
GC - Garbled Circuit
GDPR - General Data Protection Regulation
GWAS - Genome-Wide Association Study
HD - Hamming Distance
HE - Homomorphic Encryption
HIPAA - Health Insurance Portability and Accountability Act
IBE - Identity-Based Encryption
IC - Interval Containment
IS - Identification Server

LFW - Labeled Faces in the Wild
LSB - Least Significant Bit
MAC - Multiply-Accumulate
MB - MegaByte
MHE - Multi-Party Homomorphic Encryption
ML - Machine Learning
MLaaS - Machine Learning as a Service
MPC - Multi-Party Computation
MSB - Most Significant Bit
NN - Neural Network
OT - Oblivious Transfer
PETs - Privacy Enhancing Technologies
PPT - Probabilistic Polynomial Time
RLWE - Ring Learning with Errors
RNG - Random Number Generator
RNS - Residue Number System
ROC - Receiver Operating Characteristic
RSS - Replicated Secret Sharing
SIMD - Single Instruction Multiple Data
SP - Scalar Product
SS - Secret Sharing
TAR - True Acceptance Rate
TFHE - Fully Homomorphic Encryption over the Torus
TLS - Transport Layer Security
TTP - Trusted Third Party
VDP - Vector Dot Product
VPN - Virtual Private Network

Sommaire de la thèse

Introduction

Les données pourraient être étiquetées comme le pétrole du 21ème siècle. Il existe de nombreuses applications modernes alimentées par les données, allant de l'analyse de données et de l'apprentissage automatique aux algorithmes biométriques pour ne citer que quelques-unes, dont l'impact sur la société est indéniable. En effet, l'analyse de données moderne et l'apprentissage automatique (ML) ont bouleversé de nombreux secteurs du marché, allant de l'industrie du divertissement et de la fabrication (par exemple, des algorithmes de prédiction de contenu, une détection automatique des défauts), à des domaines plus sensibles tels que la santé ou l'administration publique (par exemple, la détection précoce du cancer, la poursuite de la fraude).

Cependant, le potentiel énorme de la manipulation de données est couplé à des risques élevés. Les abus et les vols de données, en particulier lorsqu'il s'agit de données personnelles, sont des préoccupations permanentes qui peuvent être atténuées en recourant à des politiques et des techniques de confidentialité (comme cela est couvert dans les législations actuelles de protection des données telles que GDPR [75] en Europe ou HIPAA [52] pour les dossiers médicaux aux États-Unis).

Ces risques sont exacerbés dans certaines applications. Les systèmes biométriques doivent s'appuyer sur un matériel sécurisé ou des parties de confiance pour détenir les données personnelles vitales pour leurs modèles de reconnaissance, et toute la manipulation de données biométriques doit suivre des règles de sécurité strictes. Les hôpitaux et les spécialistes de la santé sont privés des avantages de la formation et de l'utilisation de modèles avec un grand volume de données de patients, qui s'est avérée très efficace pour entraîner des modèles de prédiction précis traitant des problèmes complexes, par exemple, des études d'association génomique où certains gènes sont associés à des maladies telles que le cancer pour une détection précoce et une meilleure compréhension [169]. Les banques, les institutions financières et les gouvernements sont limités aux données disponibles localement pour prévenir la fraude et poursuivre l'évasion fiscale. Les modèles de détection d'images exploitatives d'enfants [238] ont besoin de données d'entraînement qui sont en soi illégales à posséder.

L'utilisation abusive de données privées non seulement impose des dommages concrets aux utilisateurs touchés, mais menace également l'adoption de ces nouvelles innovations technologiques. Aujourd'hui, les meilleures pratiques de l'industrie exigent que les fournisseurs de services protègent les données personnelles en transit et lorsqu'elles sont au repos en utilisant le chiffrement. Cependant, dans les applications où il est nécessaire de réaliser des calculs sur les données, ces données doivent être déchiffrées avant d'être utilisées, ce qui nécessite que le fournisseur de services ait accès au matériel de génération de clés. Cela expose les données à de nombreuses menaces, notamment à l'abus par des acteurs ayant des intentions malveillantes.

Dans le domaine de la cryptographie avancée, plusieurs technologies de préservation de la confidentialité visent à traiter ces problèmes. Le chiffrement homomorphe total (FHE)[109] est un schéma de chiffrement à clé publique coûteux qui prend en charge certaines opérations entre des chiffrements (généralement l'addition et la multiplication), donnant les résultats de ces opérations lors du déchiffrement. Le calcul multipartite sécurisé (MPC) couvre une série de techniques (circuits brouillés[244], partage secret [220]) qui divisent le calcul d'une fonction donnée entre plusieurs parties distinctes, de sorte que chaque partie individuelle reste ignorante du calcul global et collabore pour calculer conjointement le résultat. Le chiffrement fonctionnel (FE)[34] est un schéma de chiffrement à clé publique qui prend en charge l'évaluation de fonctions arbitraires lors du déchiffrement des chiffrements, où la clé de déchiffrement contient les informations sur la fonction à calculer et les données originales ne peuvent être récupérées qu'avec la clé de chiffrement originale. FHE et MPC peuvent également coexister dans le chiffrement homomorphe multipartite[186] (MHE), où des versions distribuées des protocoles FHE sont réalisées par un certain nombre de parties collaboratrices.

En recentrant notre attention sur le domaine de la biométrie, l'utilisation de solutions biométriques pour l'identification et l'authentification devient de plus en plus répandue dans une grande variété d'applications telles que le contrôle d'accès, les transactions financières et même les systèmes de vote. Les données biométriques, telles que les empreintes digitales, la reconnaissance faciale ou les scans de l'iris, sont uniques à chaque individu et peuvent être facilement collectées et vérifiées, ce qui en fait un moyen pratique et sûr d'identification. Cependant, la collecte, le stockage et l'utilisation des données biométriques soulèvent également des préoccupations importantes en matière de confidentialité. Les données biométriques sont des informations sensibles qui peuvent être utilisées pour identifier et suivre les individus, et si elles tombent entre de mauvaises mains, elles peuvent être utilisées à des fins malveillantes, comme le vol d'identité ou la surveillance. De plus, une fois compromise, elle ne peut pas être modifiée comme un mot de passe, ce qui en fait une vulnérabilité permanente. Il est donc crucial de développer des techniques de protection de la confidentialité qui peuvent protéger la confidentialité des données biométriques tout en maintenant la précision et l'utilisabilité du système sous-jacent. Ces techniques peuvent inclure l'utilisation de FHE, MPC ou FE pour effectuer des calculs sur les données biométriques de manière sûre et respectueuse de la confidentialité. En appliquant ces techniques, nous pouvons nous assurer que les données biométriques ne sont pas révélées en clair et que seules les parties autorisées peuvent y accéder et les utiliser.

L'objectif principal de cette thèse est de développer et de mettre en œuvre des techniques de préservation de la confidentialité pour les systèmes biométriques, de l'extraction des caractéristiques biométriques à l'identification et à l'authentification des individus. Nous utilisons des techniques cryptographiques modernes pour concevoir des protocoles sécurisés pour une large gamme de scénarios, y compris des adversaires malveillants. Nous analysons également les lacunes de leurs garanties de sécurité menant à des attaques pratiques basées sur les sorties révélées et nous y remédions avec des contre-mesures appropriées. Enfin, nous mettons en œuvre et évaluons les performances de nos solutions, en démontrant leur applicabilité à des cas d'utilisation réels tout en maintenant une haute précision et une facilité d'utilisation.

Contributions de la thèse

Cette thèse constitue une étude globale des systèmes biométriques respectueux de la confidentialité et présente un ensemble de contributions novatrices qui abordent différents aspects de la mise en place de solutions biométriques sécurisées et pratiques :

- Une nouvelle solution d'identification faciale basée sur l'appariement privé de produit intérieur à base de FE. Cette solution (i) optimise la latence en ligne pour les mêmes garanties de sécurité en modifiant les fonctionnalités des algorithmes de chiffrement et de génération de clés FE, (ii) effectue une analyse de sécurité approfondie de la fuite de donnés d'entrée à la sortie du ‘produit scalaire, y compris des contre-mesures pour contrer les attaques basées sur celle-ci, et (iii) est testée et validée dans un scénario d'appariement facial, attestant de son applicabilité pour les cas d'utilisation d'identification à usage unique.

- Un protocole de calcul à deux parties nommé Funshade pour effectuer des calculs de distance protégés par la confidentialité avec une comparaison subséquente à θ, construit à partir d'une combinaison de Secret Sharing avancé [193] et de Functional Secret Sharing [38]. Ce protocole *(i)* nécessite seulement une communication en ligne, réduisant ainsi les coûts de communication par rapport aux protocoles à deux tours existants [213, 38], *(ii)* envoie seulement deux éléments de l'anneau en ligne, réduisant la taille de la communication des solutions précédentes d'un facteur de $2l$ (pour des vecteurs de longueur l), *(iii)* possède une exactitude de 100% dans le résultat de comparaison, et *(iv)* est implémenté et open-sourced dans une bibliothèque Python autonome avec des primitifs C++ efficaces.

- Une méthode innovante pour effectuer une identification biométrique à la fois sécurisée et respectueuse de la vie privée basée sur la notion de test de groupe nommée Grote. Cette méthode *(i)* remplace les comparaisons élémentaires K par des tests de groupe pour réduire le nombre d'opérations coûteuses et non linéaires dans les calculs chiffrés, *(ii)* est instantiée et testée avec FHE et le schéma CKKS, montrant qu'elle *(iii)* a un impact minime sur la précision du système tout en accélérant son exécution de 1,5 fois.

- Un nouveau protocole de déchiffrement avec masquage collaboratif basé sur la variante multipartite [186] du schéma d'homomorphisme de chiffrement Brakerski-Fan-Vercauteren (BFV) [100] nommé Colmade. Ce protocole *(i)* effectue un déchiffrement dans un groupe de utilisateurs partagé, en les employant pour masquer un fragment du texte chiffré lors du déchiffrement tout en restant indifférent à la computation complète, *(ii)* garantit la confidentialité de tous sauf un bit de la sortie révélée dans divers modèles de menace; *(iii)* est utilisé pour construire un système d'identification biométrique respectant la vie privée et vérifiable; et *(iv)* est implémenté, testé en profondeur et mis à disposition en open-source.

- Un nouveau protocole pour l'inférence sécurisée de réseaux neuronaux binarisés basé sur le partage de secrets répliqués appelé BANNERS. Ce protocole *(i)* garantit la sécurité avec abandon contre un adversaire malveillant dans un cadre tripartite, *(ii)* a une performance équivalente aux protocoles semi-honnêtes existants.

- La conception et l'implémentation de PYFHEL, un wrapper Python pour la bibliothèque Microsoft SEAL [219], extensible à d'autres bibliothèques C++, offrant *(i)* une installation en un clic, incluant les bibliothèques back-end sous-jacentes, *(ii)* une couche d'abstraction de haut niveau axée sur Python qui facilite considérablement le travail avec FHE, y compris *(iii)* des API de haut niveau pour les fonctionnalités de bas niveau généralement non exposées. Nous montrons la convivialité de PYFHEL à la fois pour explorer et pour enseigner FHE.

La plupart de ces contributions ont été présentées dans des conférences dédiées au domaine des calculs protégeant la vie privée et publiées dans leurs actes. Nous listons ci-dessous les principales publications découlant de cette thèse:

[132] *Banners: Binarized neural networks with replicated secret sharing.* **Alberto Ibarrondo**, Hervé Chabanne and Melek Önen. Présenté à IH&MMSec2021.

[133] *Practical Privacy-Preserving Face Identification based on Function-Hiding Functional Encryption.* **Alberto Ibarrondo**, Hervé Chabanne and Melek Önen. Présenté à CANS2021.

[136] *Pyfhel: Python for homomorphic encryption libraries*, **Alberto Ibarrondo** and Alexander Viand. Présenté à WAHC2021.

[130] *Colmade: Collaborative Masking in Auditable Decryption for BFV-based Homomorphic Encryption.* **Alberto Ibarrondo**, Hervé Chabanne, Vincent Despiegel and Melek Önen. Présenté à IH&MMSec2022.

[131] *Grote: Group Testing for Privacy-Preserving Face Identification.* **Alberto Ibarrondo**, Hervé Chabanne and Melek Önen. Accepté à CODASPY2023.

[134] *Funshade: Functional Secret Sharing for Two-Party Secure Thresholded Distance Evaluation.* **Alberto Ibarrondo**, Hervé Chabanne and Melek Önen. Soumis à une conference.

Description de la thèse

Cette thèse est organisée de la manière suivante.

- Le chapitre 2 décrit les systèmes biométriques en séparant l'extraction des caractéristiques de la vérification, et décrit les exigences à imposer aux solutions biométriques respectueuses de la vie privée tout en mettant en évidence les défis existants.
- Le chapitre 3 introduit les techniques cryptographiques utilisées dans la thèse et propose une nouvelle bibliothèque Python pour améliorer l'utilisabilité du chiffrement homomorphe.
- Le chapitre 4 se concentre sur la protection de l'extracteur de caractéristiques.
- Le chapitre 5 traite de la protection des données biométriques et de la sécurisation de la vérification biométrique.
- Le chapitre 6 est consacré à l'étude des fuites de confidentialité lors de la révélation de la sortie et à l'application de stratégies de mitigation appropriées.
- Enfin, le chapitre 7 résume l'ensemble des contributions de cette thèse et discute les orientations de recherche futures.

Prérequis pour les systèmes biométriques confidentiels

Les informations concernant le modèle biométrique d'un utilisateur ou son identité sont la cible d'une pléthore d'attaques sur les systèmes biométriques [222, 6, 190], telles que la récupération du modèle biométrique en direct et des modèles de référence, la traçabilité d'un utilisateur à travers plusieurs services ou la déanonimisation de l'utilisateur correspondant à un modèle donné. étant donné que les traits biométriques ne peuvent être ni facilement modifiés (malgré la popularité de la chirurgie esthétique) ni réémis, la conception de systèmes biométriques qui contrecarrent ou même entravent ces attaques est d'une importance capitale. En effet, les organismes de réglementation tels que le Règlement général sur la protection des données de l'Union européenne (RGPD) [75] et la loi californienne sur la protection des données personnelles des consommateurs (CCPA) [108] ont récemment introduit de nouvelles réglementations de confidentialité qui exigent la protection des données biométriques. Dans ce contexte, les systèmes biométriques préservant la vie privée ont émergé comme une solution prometteuse pour aborder les préoccupations de confidentialité des systèmes biométriques, tout en cherchant à maintenir leur haute précision et leur performance.

En accord avec tout cela, nous sommes prêts à énumérer les exigences que nous attendons de nos solutions de biométrie respectueuses de la vie privée pour satisfaire :

(R1) - Confidentialité : Le système doit préserver la confidentialité de :

- Les modèles biométriques des utilisateurs inscrits (*confidentialité de la base de données de référence*). La base de données de modèles de référence constitue l'élément le plus important à protéger dans les systèmes biométriques, car elle contient les informations biométriques de tous les utilisateurs inscrits, et donc sa divulgation à un adversaire pourrait lui permettre de se faire passer pour des utilisateurs inscrits.
- Les modèles biométriques des utilisateurs en cours de vérification (*confidentialité du modèle en direct*). Bien qu'ils ne soient pas aussi importants pour le système biométrique dans son ensemble, les modèles en direct méritent également une protection, car ils contiennent les informations biométriques des utilisateurs en cours de vérification. Leur divulgation pourrait entraîner de graves préoccupations pour la vie privée de ces utilisateurs : par exemple, le suivi d'utilisateurs d'un système à l'autre (ou même d'un système à l'autre), ou la possibilité de se faire passer pour l'utilisateur.
- La confidentialité des paramètres de l'extracteur de caractéristiques utilisé par le système (*Confidentialité de l'extracteur de caractéristiques*). Les extracteurs de caractéristiques modernes sont très complexes et coûteux à développer (ressources informatiques, données requises, expertise technique), et en tant que tel, ils constituent un atout précieux à protéger, car une tentative de préserver les droits de propriété intellectuelle de son propriétaire.

(R2) - Irreversibilité : Afin de garantir qu'en cas d'attaque sur une base de données de stockage biométrique, les attaquants ne peuvent pas récupérer les informations biométriques privées réelles de l'utilisateur à travers les données stockées dans la base de données, ces transformations doivent être irréversibles. Ainsi, étant donné un modèle protégé, une entité détenant ledit modèle ne devrait pas être en mesure de récupérer le modèle biométrique original à moins qu'elle n'ait accès au matériel secret.

(R3) - Unlinkability : Rendre les informations biologiques réelles des utilisateurs non connectées de l'extérieur. Il est souhaitable d'utiliser des systèmes utilisant des données modifiées ou générées indirectement pour la vérification. Les caractéristiques biométriques réelles n'étant pas connectées aux systèmes numériques (à moins de se placer dans un scénario de type Metaverse ultra-connecté), la probabilité qu'elles soient compromises par des attaques correspondantes lancées depuis le réseau est beaucoup plus faible. Par conséquent, étant donné un modèle protégé/sécurisé, une entité détenant ledit modèle ne devrait pas être en mesure de relier le modèle biométrique crypté d'un utilisateur à son identité.

(R4) - Exactitude : Le système doit être résistant aux manipulations malveillantes pendant la phase de vérification. Il peut être possible pour un adversaire d'altérer les calculs de cette phase, et il est donc important de comprendre si le système a été soumis à une telle attaque et de fournir des contre-mesures.

(R5) - Préservation de la précision : Le système protégé doit être capable de vérifier correctement les utilisateurs valides avec une forte probabilité (faible FRR), tout en rejetant les imposteurs avec une forte probabilité (faible FAR).

Comme il s'agit d'une exigence fondamentale de tout système biométrique, leurs équivalents préservant la confidentialité doivent également respecter cette exigence. Nous nous attendons à ce que la précision du système soit raisonnablement préservée après son adaptation avec des solutions préservant la vie privée. Cette exigence découle du fait que les solutions de préservation de la confidentialité que nous envisageons ne sont pas parfaites et qu'elles peuvent donc introduire des erreurs de calcul dans les opérations.

(R6) - Préservation des performances : La version sécurisée du système biométrique ne doit pas introduire une surcharge excessive en termes de complexité de calcul ou d'exigences de stockage. En d'autres termes, elle doit présenter une latence de vérification suffisamment faible tout en étant capable de gérer un nombre suffisamment important d'utilisateurs.

En bref, le système doit être capable de s'adapter aux besoins de l'application dans laquelle il sera déployé. Ceci est particulièrement important dans le contexte des systèmes biométriques, car ils sont souvent déployés dans des scénarios à grande échelle, tels que les aéroports, où ils doivent être capables de gérer un grand nombre d'utilisateurs en peu de temps.

Résumés de chaque publication en Français

Banners: Binarized neural networks with replicated secret sharing

Les réseaux neuronaux binarisés (BNN) sont des implémentations efficaces des réseaux neuronaux convolutionnels (CNN) qui constituent l'extracteur de caractéristiques. Cela les rend particulièrement adaptés à l'inférence rapide et peu gourmande en mémoire des réseaux neuronaux fonctionnant sur des dispositifs à ressources limitées. En raison de l'intérêt croissant pour la reconnaissance biométrique basée sur les CNN sur des dispositifs potentiellement non sécurisés, ou dans le cadre d'une authentification multifactorielle forte pour des applications sensibles, la protection de l'inférence des CNN sur les dispositifs périphériques est devenue impérative. Nous proposons une nouvelle méthode pour réaliser une inférence sécurisée de BNN en s'appuyant sur un calcul multipartite sécurisé. Alors que les articles précédents offraient une sécurité dans un cadre semi-honnête pour BNN ou une sécurité malveillante pour CNN standard, notre travail offre une sécurité avec abandon contre un adversaire malveillant pour BNN en s'appuyant sur le partage de secret répliqué (RSS) pour une majorité honnête avec trois parties de calcul. Expérimentalement, nous implémentons Banners au-dessus de MP-SPDZ et le comparons avec des travaux antérieurs sur des modèles binarisés entraînés pour les jeux de données de classification d'images MNIST et CIFAR10. Nos résultats attestent de l'efficacité de Banners comme technique d'inférence préservant la vie privée.

Pyfhel: Python pour les bibliothèques de chiffrement homomorphe

Le chiffrement entièrement homomorphe (FHE) permet d'effectuer des calculs privés sur des données chiffrées, sans divulguer ni les entrées, ni les valeurs intermédiaires, ni les résultats. Grâce aux progrès récents, le chiffrement totalement homomorphe est devenu réalisable pour un large éventail d'applications, ce qui a suscité une explosion d'intérêt pour le sujet et des déploiements révolutionnaires dans le monde réel. Étant donné la présence croissante de la FHE au-delà de la communauté académique de base, il y a une demande croissante pour un accès plus facile à la FHE pour un public plus large. Les implémentations efficaces des schémas FHE sont pour la plupart écrites dans des langages à haute performance comme le C++, ce qui constitue une barrière d'entrée élevée pour les utilisateurs novices. Nous devons introduire l'FHE dans les langages (de plus haut niveau) et les écosystèmes avec lesquels les non-experts sont déjà familiarisés, comme Python, le langage standard de facto de la science des données et de l'apprentissage automatique. Nous y parvenons en enveloppant les implémentations existantes de FHE dans Python, en fournissant une installation en un clic et une commodité en plus d'une API de niveau nettement supérieur. Dans cette section, nous présentons PYFHEL, nous introduisons sa conception et son utilisation et nous soulignons comment son support unique pour l'accès aux fonctionnalités de bas niveau par le biais d'une API de haut niveau en fait un outil d'enseignement idéal pour les conférences sur l'FHE. Contrairement à d'autres travaux similaires, PYFHEL va au-delà de la simple exposition de l'API sous-jacente, en ajoutant une couche d'abstraction soigneusement conçue qui se sent chez elle dans Python.

BIOMFETRICS : Identification pratique de visages préservant la confidentialité grâce à un chiffrement fonctionnel cachant les fonctions.

En s'appuyant sur le chiffrement fonctionnel (Functional Encryption, FE) et la correspondance basée sur les produits internes, ce travail présente un système pratique d'identification de visage préservant la vie privée avec deux nouveautés clés : la commutation des fonctionnalités des algorithmes de chiffrement et de génération de clés du FE pour optimiser la latence de la correspondance (R6) tout en maintenant ses garanties de sécurité (CH2), et l'identification d'une fuite dans la sortie (CH4) pour formaliser ultérieurement deux nouvelles attaques basées sur celle-ci avec des contre-mesures appropriées[1]. Nous validons notre schéma dans un scénario réaliste de comparaison de visages, attestant de son applicabilité à des scénarios d'identification de visages à usage unique en pseudo temps réel, comme l'identification de passagers.

[1]Nous laissons l'étude de cette fuite pour le chapitre 6

FUNSHADE : Partage de secret fonctionnel pour l'évaluation de distance à seuil sécurisée par deux parties

Nous proposons un nouveau calcul bipartite, préservant la confidentialité, de diverses mesures de distance (par exemple, la distance de Hamming, le produit scalaire) suivi d'une comparaison avec un seuil fixe, qui est connu comme l'un des blocs de construction les plus utiles et les plus populaires pour de nombreuses applications différentes, y compris l'apprentissage automatique, la correspondance biométrique, etc. Notre solution s'appuie sur les avancées récentes en matière de partage de secret fonctionnel et utilise une version optimisée du partage de secret arithmétique appelée ΠSS (section 3.2.3). Grâce à cette combinaison, notre nouvelle solution nommée FUNSHADE est la première à ne nécessiter qu'un seul tour de communication et deux éléments d'anneau de communication dans la phase en ligne, surpassant tous les schémas antérieurs de pointe tout en s'appuyant sur des primitives cryptographiques légères. Enfin, nous implémentons la solution à partir de zéro en Python en utilisant des blocs C++ efficaces, ce qui témoigne de sa haute performance.

GROTE : Test de groupe pour une identification biométrique préservant la confidentialité

Cette section décrit GROTE, une nouvelle méthode d'identification des visages préservant la confidentialité, basée sur la notion de test de groupe, et l'applique à une solution utilisant le schéma de chiffrement homomorphe Cheon-Kim-Kim-Song (CKKS). Le calcul sécurisé du modèle de référence le plus proche d'un modèle vivant donné nécessite K comparaisons, autant qu'il y a d'identités dans une base de données biométriques. GROTE remplace les tests par éléments par des tests par groupes afin de réduire considérablement le nombre de ces opérations non linéaires et coûteuses dans le domaine crypté, de K à $2\sqrt{K}$. Plus précisément, nous approximons le maximum des coordonnées d'un grand vecteur en élevant à la puissance α-th et en cumulant la somme dans une disposition 2D, ce qui a un faible impact sur la précision du système tout en accélérant considérablement son exécution. Nous implémentons GROTE et évaluons ses performances.

COLMADE : Masquage collaboratif dans le déchiffrement distribué pour le chiffrement homomorphe basé sur BFV

Cette section propose un nouveau protocole de déchiffrement collaboratif pour le schéma de chiffrement homomorphe Brakerski / Fan-Vercauteren (BFV) dans un cadre distribué multipartite, et l'utilise pour concevoir une solution d'identification biométrique résistante aux fuites. En permettant le calcul d'opérations homomorphes standard sur des données cryptées, notre protocole ne révèle qu'un bit le moins significatif (LSB) d'un résultat scalaire/vectorisé en ayant recours à un pool de N parties. En employant le masquage partagé additif, notre solution préserve la confidentialité de tous les bits restants dans le résultat tant qu'une partie reste honnête. Nous formalisons le protocole, prouvons sa sécurité dans plusieurs modèles adverses, l'implémentons sur la bibliothèque open-source Lattigo et démontrons son applicabilité dans le cadre d'un scénario de contrôle d'accès biométrique.

Futur travail

Dans les chapitres 3 à 6, nous proposons plusieurs protocoles pour protéger différents aspects des systèmes biométriques. Comme dans toute proposition, ces solutions peuvent être améliorées et développées. Dans la lignée de ces travaux, des recherches futures sont envisagées pour les sujets suivants :

- Pyfhel doit être amélioré pour étendre l'ensemble des bibliothèques prises en charge (par exemple, OpenFHE [11], Helib [121]) tout en poursuivant notre travail de création d'API de haut niveau faciles à utiliser pour les fonctionnalités de bas niveau. Ce projet a gagné en popularité au sein de la communauté, ce qui a permis de créer plusieurs nouvelles fonctionnalités telles que le support partiel des clés distribuées dans BFV (DBFV) et CKKS, tout en ouvrant la porte à de nouvelles applications telles que l'analyse sécurisée des données sensibles du secteur public (par exemple, les données de santé, les informations fiscales). De plus, nous envisageons d'étendre Pyfhel pour supporter d'autres schémas de chiffrement homomorphes tels que TFHE [65].

- Les étapes futures de Banners pourraient viser les techniques de Bit Slicing pour obtenir une parallélisation considérable en tirant parti des opérations SIMD. De plus, nous envisageons l'utilisation de modèles formés spécifiquement pour l'identification biométrique (par exemple, la reconnaissance des visages). Bien que Banners soit encore lent pour la reconnaissance des visages en temps réel, il peut déjà être utilisé pour des applications biométriques sans contrainte de temps, ainsi que pour protéger un sous-ensemble de couches dans l'extracteur de caractéristiques biométriques. Enfin, nous envisageons d'étendre Banners à la formation sécurisée de BNN. Dans un même ordre d'idée, nous visualisons l'utilisation d'autres solutions basées sur des MPC non-binarisées [149] pour la protection des CNN qui composent l'extracteur de caractéristiques.

- Les protections de la vérification biométrique proposées dans cette thèse se sont jusqu'à présent concentrées sur les adversaires semi-honnêtes. Des travaux futurs pourraient étendre ces protections aux adversaires malveillants. Nous envisageons d'améliorer Funshade avec des techniques malicieusement sécurisées de SPDZ2k [77] pour les évaluations de fonctions linéaires et les extensions de FSS suggérées dans [38] pour la comparaison de seuils. De même, nous envisageons de mettre à niveau Grote avec Verifiable FHE [237] pour garantir l'exactitude de toutes les opérations chiffrées.

- L'évaluation de toutes ces solutions a été réalisée avec des données de visage provenant du jeu de données LFW [127]. Nous envisageons d'étendre nos expériences à des ensembles de données beaucoup plus importants contenant des dizaines de milliers d'identités, dans le but d'adapter les techniques de vérification biométrique à des contextes plus exigeants. De même, ces solutions peuvent être étendues à d'autres modalités biométriques telles que l'empreinte digitale ou l'iris, et même couvrir des systèmes biométriques multimodaux.

- Colmade est conçu spécifiquement pour le schéma BFV. Cependant, d'autres schémas tels que CKKS sont plus adaptés aux applications biométriques grâce au support des encodages à échelle variable (et c'est d'ailleurs la raison pour laquelle nous avons choisi CKKS pour Grote). Nous envisageons d'étendre Colmade à CKKS, et éventuellement à d'autres schémas tels que TFHE.

- Pour finir, nous avons traité séparément chacun des blocs des systèmes biométriques, de l'extraction des caractéristiques (Banners) à la vérification (BiomFEtrics, Funshade et Grote) et à la révélation contrôlée de la sortie (Colmade). La conception d'une solution de bout en bout pour les systèmes biométriques qui intègre toutes ces techniques et tient compte de leurs subtilités combinées est un problème ouvert et une direction de recherche future prometteuse.