Privacy Preserving Biometric Multi-factor Authentication

Emil Gedenryd
em3586ge-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Robert Bosch Sverige AB

Supervisors: Qian Guo (EIT)
Anders Nilson (Bosch)

Examiner: Thomas Johansson

July 1, 2023

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This thesis investigates the viability of using Fully Homomorphic Encryption and Machine Learning to construct a privacy-preserving biometric multi-factor authentication system. The system is based on the architecture described as "Model K - Store distributed, compare distributed" in ISO/IEC 24745:2022 and uses the Torus Fully Homomorphic Encryption scheme proposed by [1] to encode and compare encrypted fingerprint images. A machine-learning-based encoder is designed using the VGG11 network architecture described by [2]. The encoder is tuned for one-shot classification as a Siamese network to optimize the Euclidean distance between fingerprints from different individuals. The network is then made compatible with TFHE using the *Concrete-ml* library for Python.

Using a prototype of the system, we show that the system succeeds in preserving users' privacy with a relatively high authentication success rate. However, performance benchmarks show that the proposed encoding method is too inefficient. Finally, we highlight some areas of interest for future work that could make a system for privacy-preserving biometric multi-factor authentication viable.

Popular Science Summary

Thanks to scientific and technical breakthroughs over the last decade, applied Fully Homomorphic Encryption grows closer to maturing into a viable means to bring privacy-preserving services to the market. In this master's thesis, we investigate the viability of leveraging Fully Homomorphic Encryption in conjunction with Machine Learning to create a privacy-preserving biometric multi-factor authentication system.

In a conventional biometric authentication system, the authenticator has to be able to read and compare unencrypted biometric information during the authentication process. If the authenticator were to be compromised or dishonest to begin with, this could pose a threat to users' privacy. In this thesis, we propose a system that leverages Fully Homomorphic Encryption (FHE) to mitigate this threat.

FHE refers to encryption schemes that allow for computations on encrypted data, the results of which can then be extracted by the data owner. By encrypting fingerprint images using FHE, the authentication system proposed in this thesis is able to perform computations on and compare fingerprints without the ability to extract any biometric data belonging to the user.

Before fingerprints can be compared, they need to be encoded from an image to a numerical representation that still allows for differentiation between unique individuals. Our proposed system uses machine learning on a dedicated encoder in a process that can be described as "blind" computer vision, where an encrypted fingerprint is passed through a convolutional neural network, resulting in a tensor of encrypted values that accurately represents the individual's fingerprint. This tensor is then homomorphically compared with a previously stored tensor to decide whether or not to approve the authentication request.

By implementing a prototype of the system, we were able to evaluate it based on both accuracy and performance. Initial results indicate that the system as a whole can be viable from a security and privacy perspective. However, we found that the proposed encoding method is very inefficient in its current form, making it too slow to be used in a deployed system. Instead, we suggest that future revisions of the system remove the dedicated encoder in favour of encoding unencrypted fingerprints using the client. Further experimentation with this improvement and additional modifications could hopefully

see the proposed system become a reality in the not-too-far future.

Acknowledgements

First, I would like to thank my supervisors, Anders and Qian, for being by my side throughout my project. It would not have been possible without you. To the people at Bosch, thank you for the great opportunity to work with you and for taking me into your fold during these months.

Finally, to my friends and family, you have been my full support and number one cheerleaders right from the start. I don't know how I can repay you for your kindness. Thank you so much. I am very grateful.

List of Abbreviations

ACC Accumulator
AiTM Adversary In The Middle [Attack]
CI Common Identifier
CNN Convolution Neural Network
EER Equal Error Rate
FAR False Acceptance Rate
FHE Fully Homomorphic Encryption
FRR False Rejection Rate
GLWE General Learning With Errors
LUT Look-up table
LWE Learning With Error problem
MFA Multi-factor Authentication
NN Neural Network
PBS Programmable bootstrapping
PHE Partially Homomorphic Encryption
PI Pseudonymous Identifier
QAT Quantization Aware Training
SHE Somewhat Homomorphic Encryption
TFHE Torus Fully Homomorphic Encryption
TTP Trusted Third Party

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter presents the motivation and aims for the thesis, research questions, and an outline of the thesis.

1.1 Motivation

Multi-factor authentication (MFA) is a popular method for preventing unauthorized access to systems and user accounts. A system using biometric data as a method of authentication provides high confidence in the authenticity of the user without the need for users to memorize complicated passwords or use services that generate one-time codes.

However, the use of biometric data comes with a high responsibility for the authentication system to provide confidentiality for the data used during the authentication process. Using sound system design provides security for data in transit and during storage. However, a compromised system could pose a threat to data during the authentication process itself. Therefore, additional security features should be used to minimize the risk of users' privacy being compromised. In this thesis, we propose a solution leveraging fully homomorphic encryption (FHE) that ensures that users' privacy is preserved even if a malicious party compromises the authentication system as a whole.

Thanks to scientific and industrial efforts, the field of applied FHE has seen a lot of progress over the last decade and a half. Industry leaders predict the technology will be used in everyday applications within the next decade. Dr Randi Hindi, CEO of *Zama*, the company behind the *Concrete* and *TFHE-rs* libraries, goes so far as to state that they expect privacy-preserving end-to-end encrypted AI to be a reality within the next five year, and quote: "[...] when this happens, nobody will care about privacy anymore, not because it's unimportant, but because it will be guaranteed by design." [3].

1.2 Aim

This thesis seeks to investigate the viability of using FHE for biometric multi-factor authentication. Additionally, the project will include a prototype for a client/server architecture using hardware that emulates a real-world implementation. Once completed, the solution is evaluated based on privacy, security and

performance. The aim is also to identify areas in need of more attention to be able to create a commercial solution.

1.3 Contributions

This thesis investigates the viability of using privacy-preserving machine learning to encode fingerprints in the context of a biometric multi-factor authentication system. The system is evaluated based on performance, security, and privacy. Furthermore, alternative solutions for privacy-preserving biometric multi-factor authentication are presented and discussed.

1.4 Thesis outline

The theory used to design the proposed solution was collected using a literature review and is described in chapters 2.1, 2.5, and 2.8. The proposed system for privacy-preserving MFA is presented in chapter 3. A prototype of the system was implemented and evaluated as described in chapter 4. The results from the evaluation, along with alternate solutions, are discussed in chapter 5. Finally, the thesis's conclusions and suggestions for future work are presented in chapter 6.

Chapter 2

Background

This chapter presents the results from the literature review conducted as part of the thesis. It covers the underlying theory regarding encryption, authentication, and machine learning used in the system described in chapter 3. Additionally, the chapter presents some challenges and considerations that are related to the use of biometric data.

2.1 Encryption

On a technical level, the goal of encryption is to take an amount of data and manipulate it to make it practically infeasible to extract any information about the contents for any unauthorized parties. This is done by relying on different mathematical problems and structures that are very computationally expensive to solve for parties without access to a decryption key. The selection of underlying problem or structure will greatly impact the algorithm's security and efficiency. An example is the so-called factoring problem used by RSA, where the product of two sufficiently large secret prime numbers is used to generate private and public keys. Where, in order to break the key, an attacker would have to correctly factor the product back into these numbers, a task too expensive to perform on a conventional computer, given that a key of proper size is used [4].

Conventional encryption schemes are generally classified as either symmetric, where involved parties encrypt and decrypt information using a shared key, or asymmetric, where different keys are used for encryption and decryption, allowing for the publication of the encryption key while keeping the decryption key (referred to as private key) private. [5]

A downside of using conventional encryption for biometric authentication is that it requires the server to be able to decrypt the fingerprint before validation, as seen in Figure 2.1. In addition to being harmful to the user in case of a compromised server, the company hosting the fingerprints might face severe consequences due to privacy laws. Therefore, authentication using computations with encrypted data, made possible with FHE, could be used to ensure users' privacy.

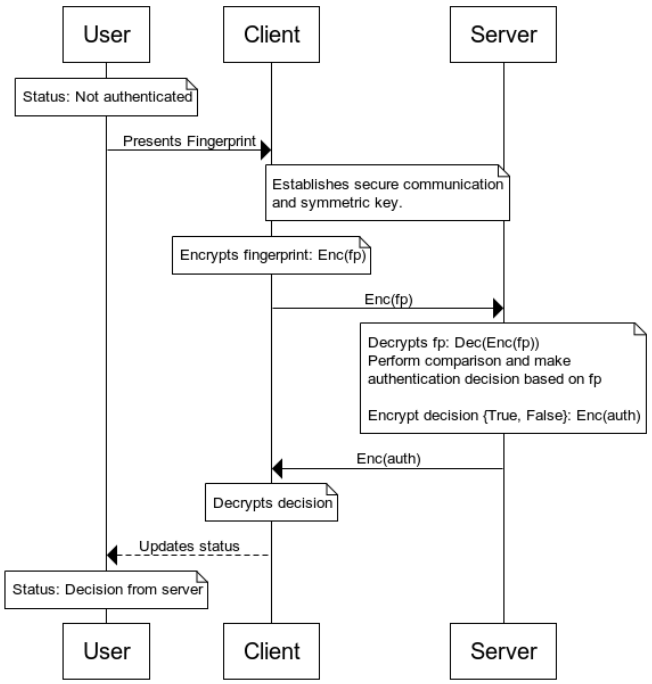

Figure 2.1: Diagram of non-homomorphic fingerprint comparison

2.2 Homomorphic encryption

The last few years have seen an increased interest in a modern paradigm of cryptography called homomorphic encryption. Schemes with this property allow varying degrees of manipulation and operations on encrypted data. Using these schemes removes previous hindrances for decentralised data, allowing applications such as cloud computing on sensitive data and machine learning as a service (MLaaS) [6].

2.2.1 Partially homomorphic encryption

Some commonly used encryption algorithms, such as the RSA and ElGamal[7] encryption schemes, have some homomorphic properties, allowing for either multiplication or addition of encrypted data, and can be classified as *Partially Homomorphic Encryption* (PHE) schemes. RSA for instance is multiplicative homomorphic for ciphertexts $\mathbf{c}_1 \leftarrow \overline{RSA}_{e,N}(\mu_1)$, $\mathbf{c}_2 \leftarrow \overline{RSA}_{e,N}(\mu_2)$ encrypted using the same parameters (e, N) as follows:

$$\mathbf{c}_3 = \mathbf{c}_1 \cdot \mathbf{c}_2 := \mu_1^e \,(mod\, N) \cdot \mu_2^e \,(mod\, N) = \mu_1^e \cdot \mu_2^e \,(mod\, N) = (\mu_1 \mu_2)^e \,(mod\, N)$$

While usable within some areas, these schemes do not allow arbitrary functionality required for more advanced applications, such as the system proposed later in this thesis [8].

2.2.2 Somewhat homomorphic encryption

Improving upon PHE, *Somewhat Homomorphic Encryption* (SHE) schemes allow for both addition and multiplication between encrypted values. However, schemes falling under this classification only allow for a set amount of operations depending on the scheme and parameters used. For example, [9] proposes an extension for the DGHV encryption scheme that allows for up to $8.54 \cdot 10^{3,457}$ additions or 9 multiplications for the worst-case scenario before plaintext correctness no longer can be guaranteed.

2.2.3 Fully homomorphic encryption

First described in the late 70s [10], the field of applied fully homomorphic encryption saw little practical use until Craig Gentry published his article on "Fully homomorphic encryption using ideal lattices" in 2009 [11]. Since Gentry's breakthrough, multiple FHE schemes have been proposed, such as CKKS[12], FHEW[13], and BGV[14]. This thesis will focus on a branch of FHE that relies on the Learning With Errors (LWE) problem, specifically the TFHE scheme described in section 2.3.

2.3 Fast Fully Homomorphic Encryption over the Torus

This thesis's work is based on the Fast Fully Homomorphic Encryption over the Torus scheme (TFHE) proposed by [1]. The scheme is based on the LWE prob-

lem and is further described in this section. The section presents some of the scheme's underlying theory as well as properties that differentiate it from other FHE schemes.

2.3.1 Preliminaries

This section covers some preliminaries for establishing the mathematical background and operations related to TFHE, such as torus mathematics and the mathematical problems that provide security to the scheme.

Torus and Torus polynomials

TFHE relies on the mathematical properties of the real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ to allow for operations on encrypted data. Such as being additive modulo 1 and multiplicative given $k \in \mathbb{Z}$, and $t \in \mathbb{T}$ as follows:

$$k \cdot t = t + ... + t \ \ (k \ times)$$

In addition to single elements, it is possible to define polynomials on the torus, allowing for additional cryptographic operations [6].

Learning With Errors (LWE) and General Learning With Errors (GLWE)

TFHE relies on the hardness of the following torus-based problems; Learning with errors (LWE) and General Learning with errors (GLWE). Where $\overline{LWE}$ and $\overline{GLWE}$ denotes the cryptographical implementation of the problems.

Definition 2.1 (LWE problem over the torus). Given $n \in \mathbb{N}$, $\mathbf{s} = (s_1, ..., s_n) \xleftarrow{\$} \mathbb{B}^n$, and an error sampled from the Gaussian distribution $\chi = \mathcal{N}(0, \sigma^2)$, the LWE problem is defined as distinguishing the following distributions [6]:

$$\mathfrak{D}_1 = \{(\mathbf{a}, r) \,|\, \mathbf{a} \xleftarrow{\$} \mathbb{T}^n, r \xleftarrow{\$} \mathbb{T}\}$$

and

$$\mathfrak{D}_2 = \{(\mathbf{a}, r) \,|\, \mathbf{a} \xleftarrow{\$} \mathbb{T}^n, r = \sum_{j=1}^{n} s_j \cdot a_j + e, e \leftarrow \chi\}$$

Definition 2.2 (GLWE problem over the torus). Given that the assumption of LWE holds, it can be extended to polynomials on the torus as follows: given $N, k \in \mathbb{N}$ with N a power of 2, $\mathfrak{s} = (\mathfrak{s}_1, ..., \mathfrak{s}_n) \xleftarrow{\$} \mathbb{B}_N[X]$, and error distribution χ over $\mathbb{R}_N[X]$, the GLWE problem is defined as distinguishing the following distributions [6]:

$$\mathfrak{D}_1 = \{(\mathbf{a}, \mathbf{r}) \,|\, \mathbf{a} \xleftarrow{\$} \mathbb{T}_N[X]^k, \mathbf{r} \xleftarrow{\$} \mathbb{T}_N[X]\}$$

and

$$\mathfrak{D}_2 = \{(\mathbf{a}, \mathbf{r}) \,|\, \mathbf{a} \xleftarrow{\$} \mathbb{T}_N[X]^k, \mathbf{r} = \sum_{j=1}^{k} \mathfrak{s}_j \cdot a_j + e, e \leftarrow \chi\}$$

Noise

The main limitation on the number of possible computations on TFHE ciphertexts is caused by the randomness introduced in the encryption process, also called noise, growing for each operation performed. For example, the sum $c_3 = c_1 + c_2$ will have the corresponding noise $e_{c3} = e_{c1} + e_{c2}$. If the level of noise in a ciphertext grows unhindered, it can cause the ciphertext to be decrypted as an incorrect value. Thus, managing the growing noise is a key problem within FHE.

A naive and cumbersome method of reducing the noise would be for the key owner to re-encrypt the data. While completely refreshing the ciphertext, this would be impractical and counterproductive in real implementations. Instead, FHE schemes use a process called bootstrapping [6].

Bootstrapping

In his 2009 article, Gentry achieved a breakthrough in the field by introducing the process known as bootstrapping, a method for reducing the noise level of a ciphertext. Reducing a ciphertext's noise enables more operations to be performed upon it before running the risk of losing information. The implementation of bootstrapping differs between FHE schemes. TFHE's implementation, called programmable bootstrapping, is described in section 2.3.5.[6]

2.3.2 Encoding/Decoding messages

Before being encrypted, messages have to be encoded to a format compatible with the TFHE scheme. This is performed using a function specified as Upper, which returns the message $\bar{x}_H \frac{p}{q}$ to the given value $\bar{x} = \bar{x}_H \pm \bar{x}_L \in \mathbb{Z}/q\mathbb{Z}$ with $0 \leq \bar{x}_L \leq \frac{q}{2p}$, as shown in equation 2.1 [6].

$$Upper_{q,p}(\bar{x}) = \frac{q}{p} \lceil \frac{p \; lift(\bar{x})}{q} \rfloor \;\; (mod\ q) \tag{2.1}$$

Where the function lift lifts elements of $\mathbb{Z}/q\mathbb{Z}$ to $\mathbb{Z}$.

2.3.3 Encryption and Decryption

Using the security assumptions of GLWE, as shown in definition 2.2, [6] defines the encryption of plaintext $\bar{\mu}$ into ciphertext $\bar{\mathbf{c}}$ using the secret key $\mathfrak{s}$ as follows:

$$\bar{\mathbf{c}} \leftarrow \overline{GLWE}_{\mathfrak{s}}(\bar{\mu}) := (\bar{a}_1, ..., \bar{a}_k, \bar{\mathfrak{b}}) \in \hat{\mathbb{Z}}_N[X]^{k+1}$$

with

$$\begin{cases} \bar{\mu}^* = \bar{\mu} + \bar{e}\,(mod(q, X^N + 1)) \\ \bar{\mathfrak{b}} = \sum_{j=1}^{k} \mathfrak{s}_j \bar{a}_j + \bar{\mu}^*\,(mod(q, X^N + 1)) \end{cases}$$

To decrypt the noisy message $\bar{\mu}^*$, the encryption process is performed in reverse as follows:

$$\bar{\mu}^* = \mathfrak{b} - \sum_{j=1}^{k} \mathfrak{s}_j \bar{a}_j \;\; (mod\,(q, X^N + 1)),$$

and extracting the message using $\bar{\mu} = \mathsf{Upper}_{q,p}(\bar{\mu}^*)$

2.3.4 Leveled operations

As previously mentioned, the purpose of using FHE schemes is the ability to manipulate the cleartext data by performing operations on a ciphertext. These operations affect the level of noise in the ciphertexts' to a varying degree depending on the operation performed.

Leveled operations are defined as operations that are performed directly on the ciphertext. This is in contrast to manipulating the encrypted data using programmable bootstrapping, as described in section 2.3.5. The TFHE is compatible with the following leveled operations:

Addition. That $\overline{GLWE}$ ciphertexts under the same key $\mathfrak{s}$ are additively homomorphic becomes apparent upon observing the encryption process: let $\mathbf{c}_1 \leftarrow \overline{GLWE}_{\mathfrak{s}}(\bar{\mu}_1)$ and $\mathbf{c}_2 \leftarrow \overline{GLWE}_{\mathfrak{s}}(\bar{\mu}_2)$ with $\mathbf{c}_1, \mathbf{c}_2 \in \hat{\mathbb{Z}}_N[X]^{k+1}$ being the respective encryption of plaintexts $\bar{\mu}_1, \bar{\mu}_2 \in \hat{\mathbb{Z}}_N[X]^{k+1}$; i.e.,

$$\mathbf{c}_i = (a_1^{(i)}, ..., a_k^{(i)}, \bar{b}_i) \; (i \in \{1, 2\})$$

with $\bar{b}_i = \sum_{j=1}^{k} \mathfrak{s}_j a_j^{(i)} + \bar{\mu}_i + \bar{e}_i$. Then

$$\mathbf{c}_3 = \mathbf{c}_1 + \mathbf{c}_2 = (a_1^{(1)} + a_1^{(2)}, ..., a_k^{(1)} + a_k^{(2)}, \bar{b}_1 + \bar{b}_2)$$

is a $\overline{GLWE}$ encryption of $(\bar{\mu}_1 + \bar{\mu}_2) \in \hat{\mathbb{Z}}_N[X]$, provided that the resulting noise keeps small.

Scalar multiplication. Given that $\overline{GLWE}$ is additively homomorphic, it follows that scalar multiplication is possible; let $K \in \mathbb{Z}_{\geq 0}$ and $\mathbf{c} \leftarrow \overline{GLWE}_{\mathfrak{s}}(\bar{\mu}) = (a_1, ..., a_k, \bar{b})$ with $\bar{b} = \sum_{j=1}^{k} \mathfrak{s}_j a_j + \bar{\mu} + \bar{e}$. Then

$$K \cdot \mathbf{c} = \mathbf{c} + ... + \mathbf{c} \; (K \, times)$$

is an encryption of $K * \bar{\mu} \in \hat{\mathbb{Z}}_N[X]$, as long as the resulting noise keeps small. For $K < 0$ then $K \cdot \mathbf{c} = (-K) \cdot (-\mathbf{c})$.

External product. $\overline{GLWE}$ does not support products between ciphertexts. Instead, TFHE relies on a matrix-based approach in the GSW construction, the general encryption of which is denoted $\overline{GGSW}$. [6] shows that by decompositioning a $\overline{GLWE}$ ciphertext $\mathbf{c}_2 \leftarrow \overline{GLWE}(\mu)$ using a gadget matrix $\mathbf{G}$, multiplication using an external product (denoted $\boxdot$) is homomorphically computable as follows:

$$\mathbf{c}_3 = \mathfrak{C}_1 \boxdot \mathbf{c}_2 := \mathbf{G}^{-1}(\mathbf{c}_2)\mathfrak{C}_1$$

where $\mathfrak{C} \leftarrow \overline{GGSW}_{\mathfrak{s}}(m_1)$ resulting in $\mathbf{c}_3 = \overline{GLWE}_{\mathfrak{s}}(0) + \mathbf{c}_2'$ $(where \; \mathbf{c}_2' \approx m_1 \cdot \mathbf{c}_2)$.

The CMux gate. By leveraging the external product, it is possible to construct a new leveled operation acting as a homomorphic selector. The 'controlled' multiplexer or CMux, takes two $\overline{GLWE}$ ciphertexts $\mathbf{c}_0$ and $\mathbf{c}_1$ that encrypts the plaintexts υ_0 and $\upsilon_1 \in \hat{\mathbb{Z}}_N[X]$, and a $\overline{GGSW}$ ciphertext $\mathfrak{C}$ encrypting a bit b. The gate outputs ciphertext **c'** encrypting υ_b according to:

$$\mathbf{c'} \leftarrow \mathsf{CMux}(\mathfrak{C}, \mathbf{c}_0, \mathbf{c}_1) := \mathfrak{C} \boxdot (\mathbf{c}_1 - \mathbf{c}_0) + \mathbf{c}_0$$

The CMux gate is an integral part of TFHE and is a key component of programmable bootstrapping.

2.3.5 Programmable bootstrapping

One of the main advantages of TFHE is its bootstrapping efficiency. It uses a combination of operations that allow the noise to be reduced while evaluating any arbitrary function using a lookup table and is therefore called programmable bootstrapping (PBS). PBS uses a secondary bootstrapping key, also known as an evaluation key, derived from the key used to encrypt the plaintext to re-encrypt the plaintext under a ciphertext with a reduced amount of noise. The bootstrapping process is made possible using the following operations:

Blind rotations. Recall that the decryption of a $\overline{LWE}$ is performed in two steps:

1. Reveal noisy plaintext as $\bar{\mu}^* = b - \sum_{j=1}^{n}$
2. Extract noise-free message $\bar{\mu} \leftarrow \mathsf{Upper}_{p,q}(\bar{\mu}^*)$

Now construct a test polynomial $\bar{\mathbf{v}}$ of degree q so that its i^{th} coefficient $\bar{v}_i = \mathsf{Upper}_{q,p}(i \; mod \; q)$. This means that the coefficient at position $\bar{\mu}^*$ will encode the noise-free value of $\bar{\mu}$. Using this fact, it is now clear that by rotating the polynomial through multiplication with $X^{-\bar{\mu}^*}$, the constant coefficient $(\bar{v}_i X^0)$ will have the noise-free value of $\bar{\mu}$, which can be easily extracted. Since we now use a polynomial, $\overline{GLWE}$ is used to enable homomorphic operations on the test polynomial. Due to the change in scheme, the ciphertext has to be rescaled to modulo 2N resulting in the approximation of the rotation factor as follows:

$$-\bar{\mu}^* \approx -\tilde{\mu}^* = -\tilde{b} + \sum_{j=1}^{n} s_j \tilde{a}_j \;\; (mod \; 2N),$$

where $\tilde{b} = \lceil \frac{2N(\tilde{b} \; mod \; q)}{q} \rfloor$ and $\tilde{a}_j = \lceil \frac{2N(\tilde{a}_j \; mod \; q)}{q} \rfloor$. Similarly, the coefficients for the test polynomials are scaled as follows:

$$\bar{v}_i = \mathsf{Upper}_{q,p}(\frac{q}{2N} i \; mod \; q) \; .$$

To perform the rotation homomorphically, we can use CMux gates and bootstrapping keys bsk[j] where:

$$bsk[j] \leftarrow \overline{GGSW_{\mathfrak{s}'}}(s_j) \; for \; all \; j = i, ..., n.$$

This allows us to iteratively rotate the polynomial homomorphically using an accumulator, essentially undoing the masking of the plaintext step by step according to:
After the rotation, the accumulator contains the $\overline{GLWE}$ encryption of

$$X^{-\tilde{b}+\sum_{i=1}^{n} s_i \tilde{a}_i} \cdot \bar{v} = X^{-\tilde{\mu}^*} \cdot \bar{v}$$

under the key $\mathfrak{s}'$, resulting in the underlying plaintext polynomial having the extractable constant term $\bar{\mu}$. Once extracted, we have a $\overline{LWE}$ noise-reduced ciphertext encrypting the plaintext $\bar{\mu}$ as shown in Figure 2.2.

Look-up Table Evaluation. By modifying the test polynomial for the blind rotation, it is possible to evaluate an arbitrary function f as part of the bootstrapping. This is done by encoding the result of $f(v_i)$ as the coefficients for the test

Algorithm 1 Blind rotation of test polynomial

$ACC \leftarrow (0, ..., 0, X^{-\tilde{b}} \cdot \bar{v})$
for $i = 1$ **to** n **do**
  $ACC \leftarrow \mathsf{CMux}(bsk[i], ACC, X^{\tilde{a}_i} \cdot ACC)$
end for

return ACC

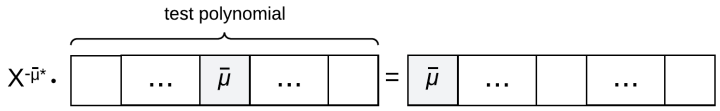

Figure 2.2: Illustration of test polynomial rotation

polynomial and using the test polynomial as a look-up table (LUT) $\bar{v}$ as illustrated by figure 2.3. The result is then extracted using the methodology behind blind rotations [6].

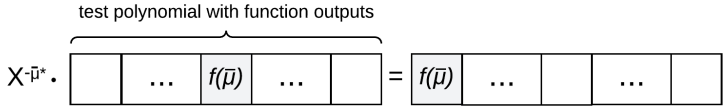

Figure 2.3: Illustration of look-up table encoded into test polynomial

2.3.6 Limitations

Even though FHE shows much promise for creating privacy-preserving applications, it still has not seen widespread use. One of the main reasons for this is the large computational overhead from the operations described in this chapter. Another is the added development complexity due to a lack of standardisation of the technology. Especially compared to other forms of encryption, which are often clearly specified by frameworks such as *RFC* [**RPCDef**]. This added complexity comes in multiple forms, such as a need for each application to implement its key and security infrastructure, something that is often inadvisable. Another obstacle is the need for the developers to be familiar with the underlying cryptography to reduce redundant overhead due to poorly chosen cryptographical parameters. Both hindrances are expected to be solved as the technology matures and becomes increasingly adopted; extensive work is already being performed on implementing efficient accelerators for optimising circuits [15].

2.4 Authentication

An essential aspect of data security is providing a means of authenticating one's peers during communication. Without a means of providing authentication's two central aspects, *identity* and *freshness*, a system is exposed to a multitude of possible attacks, such as *replay attacks*, where previously sent data is used in a manner that could compromise a system or user[5].

There are many ways of authenticating an entity; these are often grouped into the following three categories with examples:

- **Something the entity has**
  - Smart card
  - Hardware-based digital key
- **Something the entity is**
  - Biometric data (fingerprint, retinal scan)
- **Something the entity knows**
  - Passphrase
  - Cryptographic key

The different methods of providing authentication come with their own set of advantages and disadvantages, often in the form of balancing usability and security. For this reason, systems often use multi-factor authentication, where the entity must provide multiple forms of authentication to access the system.

2.4.1 Multi-factor Authentication

Multi-factor authentication (MFA) refers to using two or more authentication methods. A commonly used MFA solution combines a user password and a one-time code sent to a trusted device. By leveraging MFA, preferably using methods from the different groups of proof, it is possible to greatly decrease the risk of unauthorized access. By intentionally selecting means of authentication that complement each other, such as a pin associated with a smart card, it is possible to make the MFA seamless for the intended user while making it much harder for malicious users to exploit the system [16].

This thesis will focus on using biometric data in the form of fingerprints to provide a high level of confidence in identity for the entity requesting authorization. Combined with a method that enables confidence in freshness, such as a secure physical token, this could create a very secure system authentication source, further discussed in chapter 3.

2.5 Biometric Authentication

An intuitive method of user authentication is by using an individual's biometric characteristics, such as fingerprints or facial scans. When implemented correctly,

biometric authentication provides a means to confidently authenticate an individual without sacrificing usability. However, the use of biometric data comes with its own challenges, such as potentially increasing system complexity and requiring a higher level of trust from its users [17].

2.5.1 Choice of identifying characteristic

The choice of physiological characteristics for identification affects the system's usability and security. ISO 24745 defines desirable properties of biometric characteristics, some of which are listed below:

1. Universality: every individual should have the characteristic;
2. Uniqueness: every individual should have a distinguishable characteristic;
3. Permanence: the characteristics should not show variance over time;
4. Collectability: the characteristics should be easily collectable from the subjects;
5. Repeatability: the property of the minimization of variations of a subject's captured biometric data allowing successful recognition over time.

Additionally, the standard lists the following properties as important from an implementation perspective:

1. Performance: the success rate in recognizing individuals.
2. Acceptability: the level of willingness by the subject to use the biometric system.
3. Robustness against presentation attacks: the difficulty of using a replica of the biometric characteristic to circumvent the system.

2.5.2 Fingerprint authentication

A naive approach to fingerprint authentication could be simply overlaying two images and looking at how much they differ. In practice, the process is more complex and consists of multiple steps; preprocessing, comparison, and storage.

Preprocessing and data extraction

Before extracting any data from the fingerprint, it has to be scaled and aligned to a predetermined configuration to ensure that fingers yield a similar result each time the same finger is scanned. Once complete, the identifying data is extracted by analysing multiple features within the fingerprint, such as; the fingerprint core, deltas, pores, and more, as described by [18]. The extracted data can then be used to generate a pseudonymous identity (PI), an irreversible representation of the data, which allows the system to identify users without access to their raw biometric representation.

Comparison

A common way of comparing fingerprints is the use of mathematical distance functions. For example, the Euclidean distance, as shown in equation 2.2. Due to the elements' differences being squared, fingerprints belonging to different individuals are quickly differentiated, while smaller discrepancies caused during scanning have a lower impact on the distance.

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 ... + (x_n - y_n)^2} \tag{2.2}$$

The distance is then compared to a predetermined threshold to determine whether or not to accept the authentication request. In order to maximize the accuracy of the system, this threshold is often chosen by minimizing the equal error rate (ERR), defined as the point where the false acceptance rate (FAR) and false rejection rate (FRR) coincide. If desired, the FAR can be further reduced by lowering the threshold at the cost of increasing the risk of false rejections [19].

Storing biometric data

To ensure users' privacy, a system processing and storing biometric data should be designed to minimize the risk of compromising user data and limit the consequences if a breach happens. In addition to using PIs in place of raw biometric references, ISO 24745 suggests separating the storage of user data. For example, PIs should be stored under a Common Identifier (CI) in place of users' real identities. Furthermore, the entity responsible for storing PIs should then not have access to the mapping between CIs and user identities [17].

2.6 Privacy

Defined by the Cambridge Dictionary as "the right that someone has to keep their personal life or personal information secret or known only to a small group of people" [20]. User privacy is vital to keep in mind when creating a system for biometric authentication, not only from the users' perspective but also from a legal standpoint.

While improving system usability by removing the need for users to remember passwords or obtain secure tokens, improper handling of biometric data can result in dire consequences. For instance, if a direct representation of an individual's biometric data were to leak, it could universally compromise the use of that characteristic. For example, if a raw representation of an individual's fingerprint were to leak from service "A", it might be possible for a malicious party the use it in service "B". An encoded version of the biometric reference, known as a pseudonymous identifier (PI), is commonly used to combat this risk. In addition to potential security issues, this would be a severe breach of the individual's privacy and could increase the risk of them being the target of crimes such as impersonation or identity theft [17].

2.7 Security aspects

Several aspects are listed below to consider when designing any authentication system and even more so when it relies on biometric data. Some of these aspects are covered by ISO 24745. If these are not fulfilled, it could lead to a reduction in system integrity or the compromise of user data.

1. Confidentiality: ensures data secrecy and is referred to as the "classical" security aspect provided by encryption [5]. Not only for data in transit but also when stored, reducing the damage in case of a breach.
2. Integrity: refers to that the data has not been altered from its intended state. In the case of biometric data, it ensures that biometric references are trustworthy and usable for comparison.
3. Renewability and revocability: help protect the system in the case of compromised data. This is done by providing the ability to remove (revocability) authorization for an entry and ensuring that the sample can be used again to create a new uncompromised PI (renewability).
4. Availability: ensures authorized parties can perform the steps necessary to access the system using their biometric data. For instance, by protection against DDoS attacks.

2.8 Machine Learning

Machine learning, specifically neural networks, can be used as an alternative to FHE-incompatible PI generators such as FingerCodes, which would require the encoder to perform comparisons between encrypted values. This section presents the theory used to design a neural network-based solution for generating PIs from fingerprints encrypted under TFHE.

2.8.1 Neural Networks

Originally modelled as an analogy to the human brain [6], neural networks (NN) are composed of multiple interconnected neurons. A *neuron* refers to a nonlinear, parameterized, bounded function, illustrated in Figure 2.4. The neuron takes n signals as input, which are then weighed and biased before being passed to a nonlinear activation function f, such as $ReLU$ shown in equation 2.3. The neuron's output signal is, therefore, the value $y = f(s)$ of the activation function where $s = \sum_{i=1}^{n} w_i x_i + b$ is the weighted sum of the inputs, with the additional constant bias b [21].

This thesis focuses on a particular type of neural network called *feedforward networks*. Mathematically, feedforward neural networks are defined as a nonlinear function of its inputs, which is the composition of its neurons. This type of network is represented as a set of neurons connected together, as illustrated by Figure 2.5 [21].

$$ReLU(x) \;=\; \max(0, x) \tag{2.3}$$

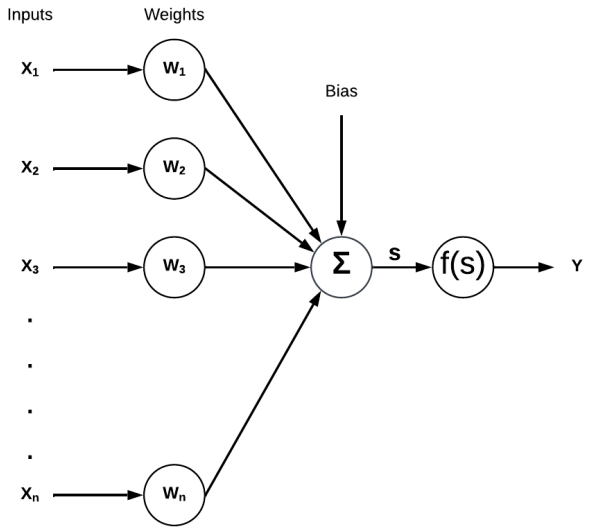

Figure 2.4: Illustration of a neuron

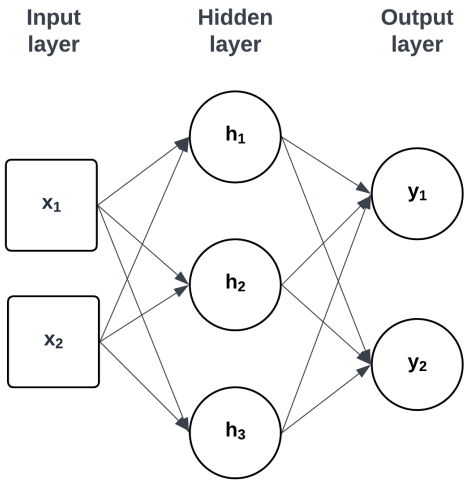

Figure 2.5: Illustration of a feedforward neural network consisting of; an input layer accepting input of length two, a hidden linear layer with three neurons, and an output layer with two neurons.

2.8.2 Convolution layers

State-of-the-art neural networks within computer vision leverage convolution layers to solve problems such as object recognition and lung cancer prediction [22]. Neural networks using this type of layer, known as Convolution Neural Networks (CNNs), have been proven effective for fingerprint encoding as shown by [23], and will be used to extract features from encrypted fingerprints.

Convolution layers focus on the use of trainable kernels, which are passed over the input. For each step, the kernel's centre is placed over the input, which is then used to calculate a weighted sum used as output as shown in Figure 2.6.

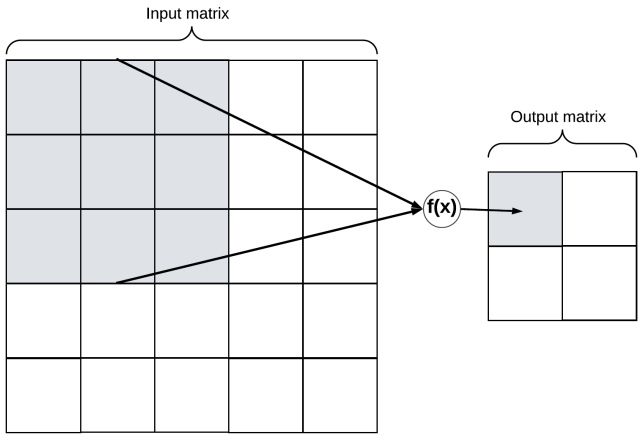

Figure 2.6: Illustration of a convolution layer

2.8.3 Training

The versatility of NNs comes from the ability to train them using data similar to their intended use case. The training data is passed through the network over multiple iterations or "epochs". The difference between the output and intended result, referred to as loss, is then used to adjust the weights inside of the network according to a predetermined loss function, such as *log-loss* [24].

2.8.4 Transfer learning

Instead of using a large amount of resources to train a network from scratch, it is possible to use a pre-trained network as a base for a neural network. This pre-trained network, such as VGGNet [25], can be altered and fine-tuned to fit the desired application.

2.8.5 Siamese networks and triplet loss

Unlike regular networks, where the output from a single data point is used in training, Siamese networks calculate the loss using the result from multiple inferences

using the same weights, as shown in Figure 2.7. This enables the network to be specialised in new ways, such as maximizing the difference between fingerprints belonging to different individuals. This specialization can be done by training the network using what is known as Triplet Loss [26]. This loss function takes the output from three different input data points. It minimizes the difference between the so-called anchor and a positive datapoint (from the same class or individual) while maximizing the difference between the anchor and a negative datapoint.

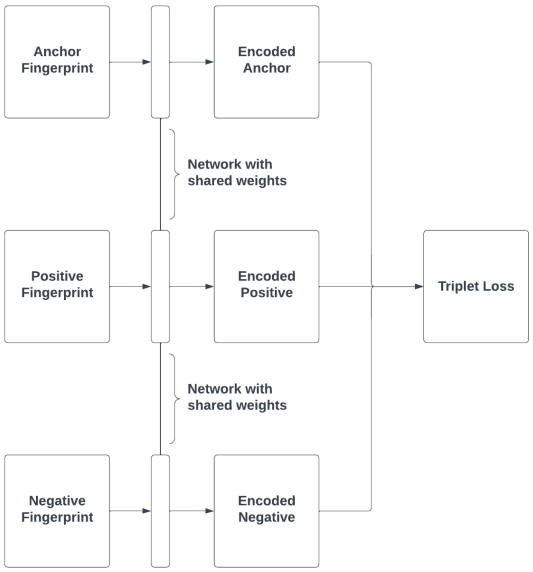

Figure 2.7: Illustration of Siamese network

2.8.6 Machine learning for fingerprint validation

Neural networks provide an FHE-compatible solution for fingerprint encoding. The components to make this possible are discussed in this subsection.

One-Shot Classification

A normal use case of neural networks is classification, where the network selects a so-called ”class” depending on the input data, for example, classifying what type of animal is present in an image. If a network were to use normal classification for fingerprint validation, eg. by outputting the individual whose fingerprint is used as input, the network would have to be rescaled and retrained for each new individual enrolled in the system. Due to this, normal classification is not a viable solution for validating fingerprints. Instead, so-called one-shot classification can be used.

A network for one-shot classification is trained using data from multiple different classes, such as fingerprints, and outputs a tensor containing the encoded

data. Using Siamese networks, the network is trained to classify identifying features of individuals and encode them into a tensor that is as unlike those from other individuals as possible [27].

Comparing Fingerprints

Once a one-shot classifying network has generated an encoding in the form of a tensor, this output can be used to compare fingerprints using one of many mathematical definitions of distance. ISO 24745 suggests using either the Hamming or the Euclidian distance between the encoded fingerprints as a simple comparison metric.

Choice of threshold

Due to the threshold being a critical factor in the system's security, it is vital to choose it accurately. One way to do this is by calculating the equal error rate (EER), which is the point where it is as likely for the system to reject an authorized user (FRR) and accept a non-authorized user (FAR). Depending on the system's security requirements, it would be possible to make the threshold even lower than the EER, reducing the risk of unauthorized access at the cost of overall usability [19].

2.8.7 Machine learning using TFHE

A big advantage of TFHE is the ability to infer encrypted values as described by [6]. This is possible using weights stored in the clear, resulting in linear and convolutional layers evaluated using levelled operations. Evaluation of activation layers can then be performed using programmable bootstrapping, thus resetting the noise to a manageable level and allowing networks to have theoretically limitless depth. While inference time increases when performing inference on encrypted inputs, [6] shows that networks do not suffer any major loss in accuracy when properly adapted to TFHE. Adapting the network refers to making it compatible with TFHE by only using integers as weights using quantization. This also ensures that the 16-bit value limit is not exceeded [28].

Chapter 3
Method

We propose a system allowing for privacy-preserving multi-factor authentication that leverages TFHE for encoding and comparison between fingerprints. The system is designed with ISO 24745 taken into consideration and provides confidentiality, renewability, irreversibility, and unlinkability for users' biometric data.

This chapter describes the design of the system and the privacy-preserving components that make it work.

3.1 Architecture

The system uses a client-/server architecture based on a model suggested in ISO 24745 as "Model K - Store distributed, compare distributed"[17]. Figure 3.1 shows an overview of the suggested architecture. The server consists of subsystems for encoding, authorization, and storing fingerprints. The client, on the other hand, is responsible for collecting the users' fingerprints and keys. These keys would be stored on a secure external token exclusive to each user, such as an access card or car key fob. Communication will be protected using the *Signal* protocol as described by [29] to prevent malicious parties from taking advantage of the system. This protocol, used by applications such as *WhatsApp* and *Facebook Messenger*, protects communication from common attacks like AiTM and replay attacks. In addition to protection against active attacks, Signal provides both forward and backward secrecy in case a key is compromised.

3.1.1 Client

The architecture for the client is relatively flexible. The main requirement is that the CNN used by the encoder has been trained using images from the same type of fingerprint sensor used by the client. Otherwise, differences in resolution or size could cause erroneous encodings. In addition to compatibility with the system, it is important to ensure the hardware is secure against malicious use, such as tampering or spoofing attacks.

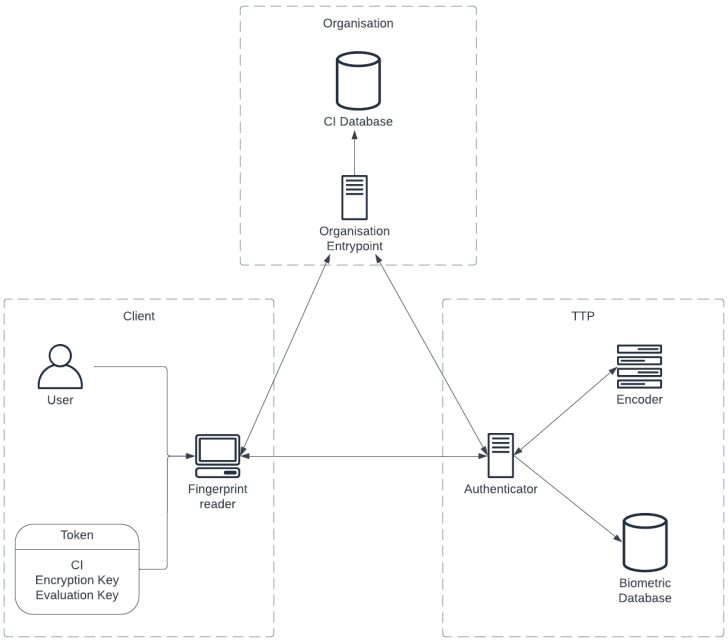

Figure 3.1: Overview of proposed architecture

3.1.2 Server

The authentication service would consist of three subcomponents, an encoder, an authenticator, and storage. On top of these, some extra infrastructure is needed to facilitate the system, such as a key server for use by the signal protocol. However, these fall outside of the scope of the thesis.

Encoding encrypted fingerprints

By leveraging TFHEs Programmable bootstrapping, a CNN can be trained to encode encrypted fingerprints using one-shot classification. This network would preferably use transfer learning from a well-established CNN architecture, such as VGG11 [2]. The pre-trained network could then be tuned as a Siamese network with triplet loss to generate encrypted tensors of a set size to represent the fingerprint.

In the proposed system, the fingerprints are encoded using a dedicated encoder as part of an external trusted third party (TTP). The encoder uses encrypted fingerprints along with the corresponding evaluation key to encode the fingerprints without gaining access to either the biometric data or the identity of the user. Once the fingerprints are encoded into an encrypted tensor using the CNN, they are sent to the authenticator for either enrollment or authentication.

Comparing encoded fingerprints

Using the fact that fingerprints are encoded to minimise the Euclidean distance for fingerprint images from the same individual, the same distance function can be used during comparison. Since this function can be implemented using a TFHE circuit, encoded encrypted fingerprints can be compared, provided they are scanned using the same type of sensor and encrypted under the same key. Using a distance threshold calculated during training, this comparison can then be used to authenticate users in the system.

3.2 Enrollment

In a non-privacy-preserving biometric system, the enrollment phase is relatively straightforward. The enrollee presents their biometric characteristic to a sensor and their identifying data, such as a user ID. The processed biometric data is then stored along with the identifier for use in the authentication phase.

In the case of a privacy-preserving system, the process becomes slightly more complex as a result of keeping the users' identities as hidden as possible. In addition to preserving privacy, a user needs to enrol their fingerprint encrypted under the same key used during authentication due to fingerprints being compared homomorphically. This means the users' encryption keys must be distributed or generated before the enrollment can proceed. From an implementation perspective, the easiest solution would be for new users to physically go to a location, such as the organisation owning the system, where their fingerprint is scanned and token handed out. While usable on a small scale, for instance, within an organisation's internal access control, this solution can run into scalability and usability issues.

Instead, the system proposed by this thesis uses the enrollment procedure shown in Figure 3.2. The organisation generates a CI for the user, which the TTP uses as the only user identification method. Once the client has received the CI, the user's fingerprint is scanned and encrypted using their encryption key. The encrypted fingerprint is then sent to the TTP with the CI and evaluation key. To reduce the network bandwidth used by the client during the authentication phase, the evaluation key can be stored on the server. This would additionally remove the need for the evaluation key to be stored on the user's token, thus reducing its hardware requirements.

3.3 Authentication

During the authentication phase, the client scans and encrypts the user's fingerprint. This is then sent with the user's evaluation key to a dedicated encoding service equipped with the encoding network. The encoder then generates an encrypted tensor without the ability to read the user's plaintext fingerprint. This tensor is then sent to the authenticator. The authentication decision is then made based on the homomorphic comparison between the session fingerprint and the enrolled fingerprint stored in the database. In order to prevent so-called hill-

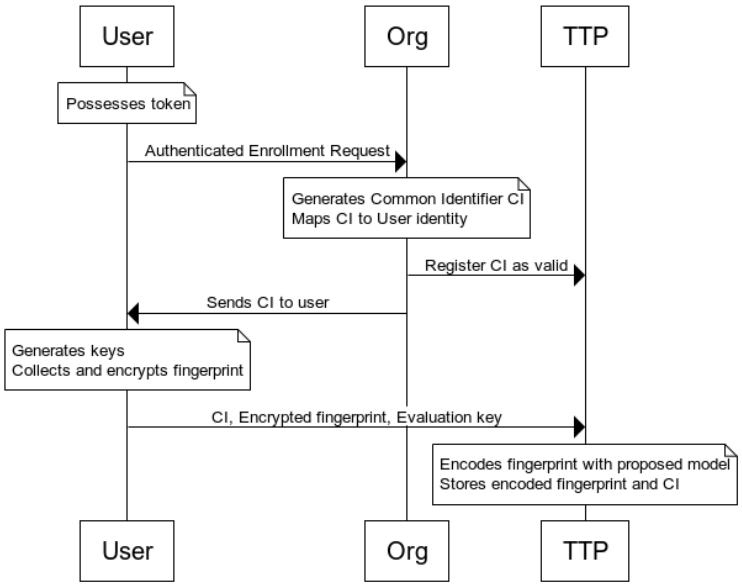

Figure 3.2: Diagram showing enrollment phase

climbing attacks, the authenticator returns the decision, not the distance between the fingerprints, to the client, as suggested by ISO 24745. An overview of the authentication phase is presented in Figure 3.3.

3.4 Revoking fingerprints

Per ISO 24745, the system allows the system owner or the user (via the system owner) to initiate a process to revoke authorization privileges, as shown in Figure 3.4. This includes not only the removal of the CI from the list of authorized users but also removing all biometric and identifiable data. According to ISO 24745, the data will be removed from active and archived services and all backups. Depending on the reason for revoking a user's authorization, such as removing a user, any other record of the user, including mapping between CI and identity, should be removed.

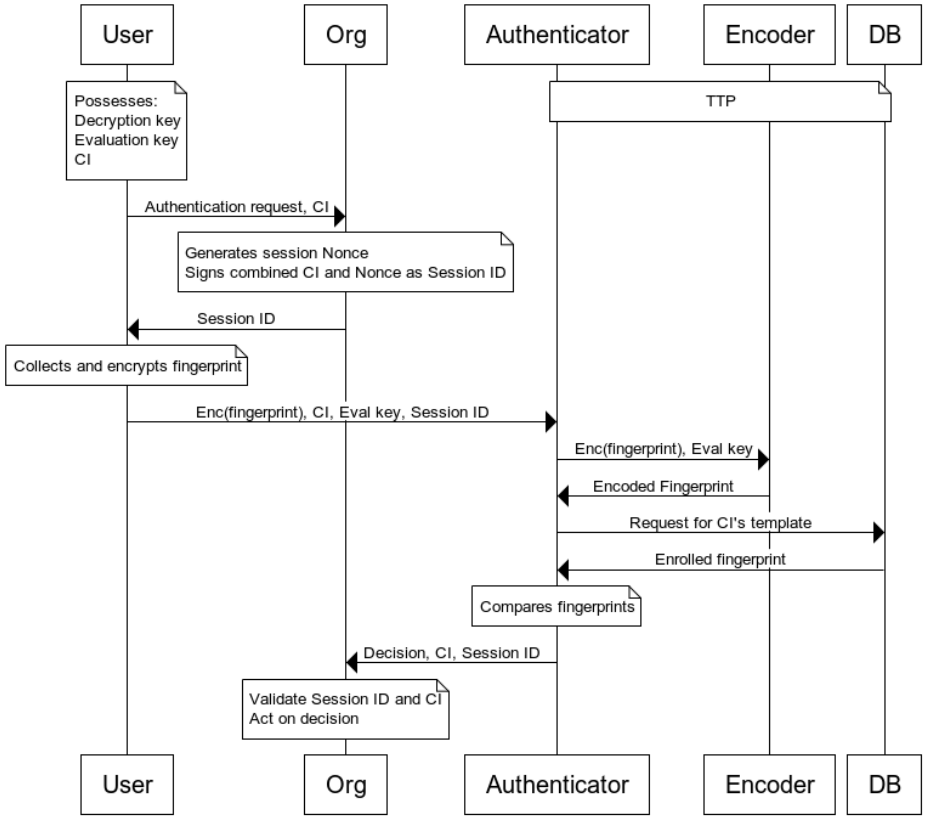

Figure 3.3: Diagram showing authentication phase

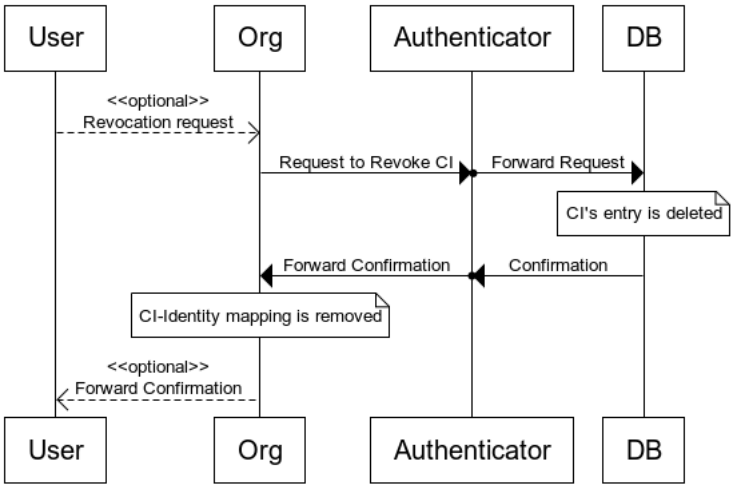

Figure 3.4: Diagram showing the revocation process

Chapter 4
Implementation and Results

In order to investigate the practical feasability of using FHE in the context of MFA, a prototype was implemented and evaluated as described in this chapter.

In addition to the features implemented for the prototype, some alternate solutions are discussed in section 5.2.3.

4.1 Scope and Limitations

Due to a lack of maturity in the field of applied FHE, some compromises have been made for the prototype's implementation. To ensure the validity of our results and conclusions, complementary measurements will be taken in addition to those generated using the prototype.

4.1.1 Encoder

TFHE-derived limitations

Due to the large performance overhead of TFHE and Concrete's limitation to 16-bit integers, the neural network used in the encoder is run with Concrete's so-called "*Unsafe features*". This allows the network to be run using a simulated circuit, reducing computation time drastically. Additionally, enabling the unsafe features allow for experimentation with values that would otherwise cause fingerprint comparison to fail due to the limit on integer values. However, some limitations are still put upon the networks' size and output values in order to approximate the architecture for a real-world use case.

4.1.2 Authentication system

Fingerprint storage and encoding

Due to the encoder being run using a simulated circuit, fingerprints are stored in plaintext. This allows the prototype to use pre-encoded fingerprints. These fingerprints are instead encrypted before comparison by the authenticator. The compromise does not affect the authentication phase, which will remain the focus of the evaluation.

Key management

Due to fingerprint encodings being pre-generated for the prototype, the client doesn't use any TFHE keys. Instead, fingerprints are encrypted at the time of comparison with the help of Concrete's built-in key generation. This results in a key management system not being required.

During the implementation of the prototype, Concrete's built-in key generation is used on a session-to-session basis. Therefore, no persistent key management system is used.

4.2 Prototype Implementation

4.2.1 Encoder

An example provided by Zama on image classification and network fine-tuning was used as a basis for the prototype. The example makes use of a slightly modified VGG11 network, as well as a QAT-compatible network for use in FHE. Some modifications were made to the network for use in the prototype. Namely an increase of the networks' input- and output sizes to allow for higher resolution fingerprint images, and to increase the separation between encoded fingerprints. In addition to the change in sizes, the fingerprint encoding is clamped to limit integer values during comparison under FHE.

Dataset

The encoder was trained using the *L3-SF Database* [30] synthetic fingerprint dataset. Each of the five subsets contains ten images, as shown in Figure 4.1, from 148 unique individuals. Two subsets were used during the construction of the encoder, one for training and the other for validation. Due to the dataset containing multiple images per individual, it is possible to use triplet loss when training the network.

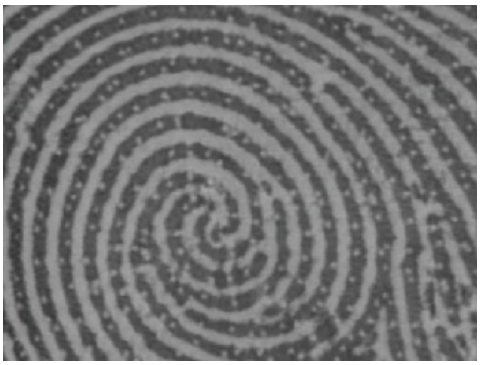

Figure 4.1: Unedited example image from the L3-SF Database.

Training

The network was initially pre-trained using the non-quantized VGG11-like architecture displayed in table 4.1 over two phases. Firstly to train the fully-connected final layer and then the whole network, until a satisfactory performance had been reached. This was done using the *PyTorch* library for Python. The pre-trained network was then quantized using the libraries *ONXX* and *Brevitas*, to make it compatible with homomorphic operations. After quantization, a shorter second round of training was performed to recuperate the loss in accuracy from quantization.

Layers
Input image - 240x320
Conv2D - 64
AvgPool2D
Conv2D - 128
AvgPool2D
Conv2D - 256 Conv2D - 256
AvgPool2D
Conv2D - 512 Conv2D - 512
AvgPool2D
Conv2D - 512 Conv2D - 512
AvgPool2D
Flatten
FC - Embedding length

Table 4.1: Network architecture used by encoder

4.2.2 Authentication system

Using the functionality of the encoder, the surrounding infrastructure is trivial to implement. It contains an authenticator for comparing fingerprints, a database for storing enrolled fingerprints, and a client connecting to the services.

Client

The prototype uses a modified version of the client proposed in section 3.1.1. Instead of using a physical token and a fingerprint reader, the CIs are generated on the client before fetching encoded fingerprints directly from the encoder. This results in an altered communication flow for the enrollment and authentication phases, as shown in Figures 4.2 and 4.3.

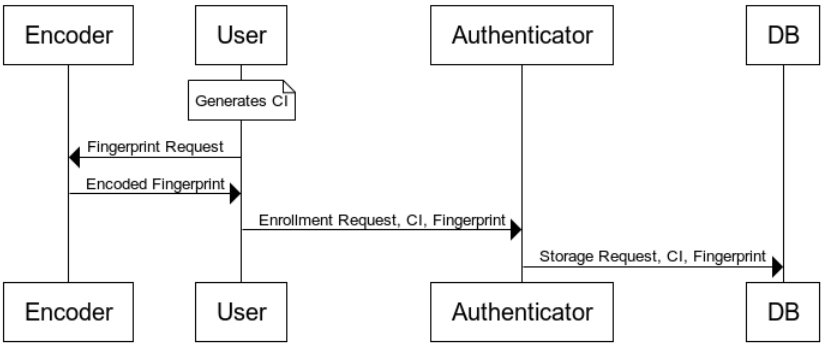

Figure 4.2: Diagram showing the modified enrollment phase used in the prototype.

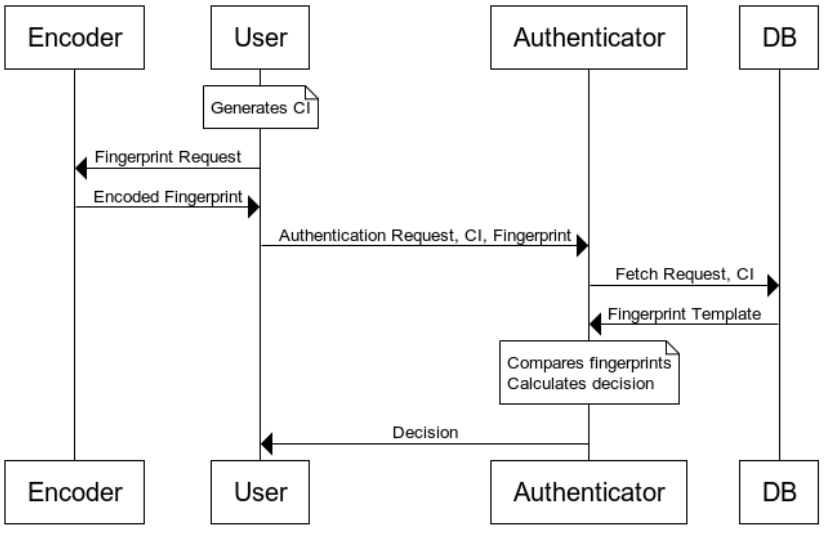

Figure 4.3: Diagram showing the modified authentication phase used in the prototype.

Authenticator

Using a compiled TFHE circuit for calculating the Euclidean distance between two encrypted tensors, is capable of comparing encrypted encoded fingerprints. This comparison is then used to generate a decision based on the predetermined threshold. The authenticator is also responsible for querying the database for a previously enrolled fingerprint. To prevent malicious exploration of the system, any query made by a client for authentication against a non-existing fingerprint will be met with a message as if the authentication failed. Thereby preventing a malicious client from testing whether or not some hash belongs to an enrolled user.

Database

The final service in the system is the database used for storing enrolled fingerprints. A regular database is used for the initial prototype, storing fingerprints under each user's CI. Improvements could, however, leverage FHE-based databases as proposed by [31].

Communication

Following the proposed design, the prototype uses the Signal prototype to secure communication. This mitigates the risk of replay- and AiTM attacks. While this comes with some overhead and would require additional infrastructure in a production environment, the benefits from the added security outweigh the costs.

4.3 Results

In this section, measurements from the prototype are presented. First, performance benchmarks for both the TTP and the client are evaluated. Finally, an evaluation of threats and countermeasures related to the proposed system's privacy and security is presented.

4.3.1 Performance evaluation

This section presents the results of the performance measurements collected from the different components of the prototype. Component performance measurements exclude overhead from communication with other parts of the system unless explicitly stated. Table 4.2 shows the hardware used for running evaluations.

Encoder

The benchmarks for the encoder were run on an otherwise idle machine, allowing the CNN to use resources freely. Table 4.3 shows the accuracy, time, and memory usage for encoding fingerprints. Since Concrete-ml lacks GPU support, the model for encoding encrypted fingerprints is simulated on the CPU, while the Torch model is evaluated on both the CPU and the GPU. The model's accuracy is measured using the threshold achieving EER. Memory usage shows each model's

Model	CPU	GPU	Memory	Role
HP Z2 G9	Intel Core i7-12700k	Nvidia Quadro T400	32 GB	TTP
Raspberry Pi 3B+	ARM Cortex-A53	-	1 GB	Client

Table 4.2: List of hardware used for evaluation.

average virtual memory usage; note that the CPU was the limiting resource for both models not running on the GPU.

Model	Accuracy [%]	Time [s]	Memory [GB]
Torch (CPU)	94	0.2	8.0
Torch (GPU)	94	0.03	13.2
Simulated TFHE (CPU)	93	517	23.9

Table 4.3: Performance measurements for the encoder.

Authenticator

Due to intermediate values exceeding Concrete's bit width limit when computing the Euclidean distance, there is an upper limit to encoding size for the prototype. Measuring the distance between two fingerprints using smaller encodings yields the results seen in Table 4.4.

Encoding Length	Maximum Value	Bit width	Time [s]	Accuracy [%]
13	4	8	9	83
13	16	12	19	86
13	64	16	23	90
128	255	23	-	93

Table 4.4: Authentication performance for different encoding sizes.

Client

Attempts to encrypt images using the Raspberry Pi specified in table 4.2 proved unsuccessful due to the limited available RAM. Instead, encryption performance was measured on the desktop specified in table 4.2. Table 4.5 shows the measured memory usage and time spent encrypting images of full, half, and quarter resolution. The memory usage reported in table 4.5 shows the increase in allocated virtual memory in addition to the 2.5GB of memory allocated when not performing the encryption.

Resolution [pixels]	Memory [GB]	Time [s]
320x240	1.9	5.70
160x120	0.9	2.86
80x60	0.5	1.43

Table 4.5: Image TFHE encryption benchmarks.

4.3.2 Privacy and security evaluation

This section presents the evaluation of the proposed system from a privacy and security perspective. The evaluation is performed using the requirements defined in ISO 24745. The results in table 4.6 show potential threats to the system, the related privacy and security aspects, and countermeasures to mitigate the threat.

Brief description of threats and countermeasures

For clarity, brief descriptions of select threats and countermeasures are provided as follows:

- Adversary in the middle (AiTM) refers to a malicious party intercepting packets with the ability to read, insert and modify the biometric data in transit without either party knowing that the established link has been compromised.
- Encryption key disclosure would allow an attacker to decrypt a user's PIs if a PI database becomes compromised. Due to PIs being unlinkable and irreversible, this scenario would not compromise the user's raw biometric data.
- Hill climbing attack refers to a malicious systematic modification of the scanned fingerprint to obtain progressively lower distance until the decision threshold is met.
- Channel security is provided using the Signal protocol. Provided that a secure configuration is used, this gives the system protection against replay attacks, AiTM, and eavesdropping.

Threat	Privacy-/Security aspect	Countermeasure
AiTM	Confidentiality, Integrity, Authenticity	Secure channel
Brute force	Authenticity, Confidentiality	Use of strong encryption
Disclosure of PI database	Confidentiality, Irreversibility, Unlinkability	Data separation, Revocable and renewable biometric references
Eavesdropping	Confidentiality	Secure channel
Encryption key disclosure	Confidentiality, Authenticity	Revocable biometric references, Irreversability, Unlinkability
Fingerprint spoofing	Authenticity	Presentation attack detection
Hill climbing attack	Authenticity	Comparison under FHE
Replay attack	Authenticity	Secure channel
Unauthorized removal or modification of PIs	Integrity	Database access control, Appropriate recovery procedures

Table 4.6: Threats and countermeasures related to privacy and security.

Chapter 5
Discussion

This chapter discusses the results gathered from evaluating the system prototype based on performance, security, privacy, and practicality. It then brings up the method and decisions made when implementing the prototype. Finally, privacy-preserving MFA is discussed in a wider context.

5.1 Results

The results show that creating a secure and privacy-preserving biometric MFA system with PIs generated from encrypted fingerprint scans is possible. While fingerprint comparison can be performed relatively efficiently, it is obvious that the encoding method used in this thesis is too inefficient to be practical in an actual implementation. This could be solved using an alternative encoding method in the clear on the client, discussed in section 5.2.3. However, it might be possible to retain the ability to encode encrypted scans using one or more of the following suggested solutions:

1. Improve the performance of TFHE operations: In its evaluated state, the Concrete library has limitations that greatly reduce the efficiency of any machine learning model. However, regular performance improvements are being brought to the library that could reduce the encoding time using the current solution. One feature that would greatly impact the performance of the encoder would be the ability to run the inference on the GPU.
2. Use of an alternate encoder architecture: Since this thesis aimed to investigate the possibility of implementing privacy-preserving MFA, the architecture for the encoding network was not chosen based on efficiency. Instead, VGG11 was chosen because it was proven to be migratable to Concrete without any major loss in accuracy. Therefore, an encoder using an alternate network architecture could improve the encoding efficiency.
3. Reduce fingerprint scan size: Due to the legal restrictions related to working with biometric data, the choice of a dataset for network training was very limited. This led to the network being trained on 320x240 pixel images. To ensure the validity of this thesis's results, the images were not compressed or cropped, resulting in the network having a much larger input than might be necessary. For reference, the example used as a basis for the encoder uses

a very similar architecture to simulate FHE inference on images with a size of 30x30 pixels. This takes them 38 seconds per inference, compared to our 517 seconds.*

5.2 Method

This section discusses the method used to implement and evaluate the prototype. This includes hardware for evaluation, the dataset used for network training, and alternate encoding methods.

5.2.1 Hardware

The hardware used to evaluate the prototype was chosen based on the processor architecture to ensure that the solution would be compatible with a use-case envisioned by Bosch. However, this does not mean that the execution times measured during evaluation are representative of running the components on its envisioned hardware. Instead, execution times should be viewed as an approximation of a real-world use case and an indicator of the computational overhead caused by running inference on encrypted data.

5.2.2 Dataset

As discussed in section 5.1, the dataset used in this thesis was chosen due to legal restrictions on non-synthetic fingerprints. This resulted in the encoder being evaluated on relatively homogenous data. To perform a more thorough evaluation of the network, it would be beneficial to train and evaluate it using a larger number of datasets containing fingerprints with higher variations in scan quality.

With access to more datasets, it would be interesting to investigate the possibility of training a network capable of encoding fingerprints taken from different types of sensors. If successful, this would remove the need for organisations to use pre-approved sensors. Thus, making organisations more likely to be willing to adopt the system.

5.2.3 Alternate encoding method

Due to the large overhead when encoding fingerprints homomorphically, it is worth considering an alternate solution where fingerprints are encoded in the clear using the client. As long as the encoding can be compared homomorphically and is not too computationally expensive, it could be used with the rest of the system proposed by this thesis. The most straightforward solution would be to use the PyTorch implementation of the CNN used in this thesis. More advanced network architectures could be possible if the client has enough computational resources. Using more advanced and specialised networks could lead to higher authentication accuracy. Additionally, encoding the fingerprints in the clear would allow the client

*These benchmarks were performed on different hardware and are only meant to serve as a basis of reference.

to use non-ML-based encoding methods. For example, [32] proposes a solution based on Fingercodes and the Pailler cryptosystem. Modernising the Fingercode-based solution could serve as a basis for future versions of the system proposed in this thesis.

The downside of generating an encoding in the clear is that it removes the possibility of distributing PI generation without compromising security and confidentiality, as the user's raw fingerprint scan would be made available to an external party. This could result in raw biometric data becoming compromised if the encoder were to be breached.

5.3 The proposed system in a wider context

The work presented by this thesis could be adapted for use in various real-world systems and applications. It could, for example, serve as a method for providing multi-factor authentication in domains where one-time keys are impractical or impossible to implement.

The proposed design also provides a method for organisations to reduce their security management burden by distributing the authentication process to a trusted third party. Furthermore, with limited user information being sent to the third party, the organisation can guarantee that no identifiable or unencrypted biometric information will be shared with external parties.

Chapter 6
Conclusions and Future Work

In this thesis, we implemented a CNN using transfer learning, capable of encoding fingerprints encrypted using TFHE. Additionally, a system for privacy-preserving biometric MFA was designed. The system ensures users' privacy and biometric data confidentiality in the case of a data breach in accordance with ISO 24745.

By implementing and evaluating a prototype of the system, we show that the proposed solution is viable from a privacy and security perspective. However, the benchmarks show that the system is very inefficient from a performance perspective, primarily due to the overhead caused by the inference using homomorphic operations. We propose three solutions to this problem:

1. Improve the performance of TFHE operations
2. Use an alternate encoding solution
3. Reduce the size of fingerprint scans

Additionally, the results show that the client would need a non-trivial amount of computational resources in the proposed system despite not needing to encode fingerprints. Therefore, we conclude that a solution where fingerprints are encoded on the client before encryption is preferable to the method investigated in this thesis, as discussed in section 5.2.3.

6.1 Future work

Due to the encoder being the biggest bottleneck for the proposed solution, this would be an interesting area of future work. Chapter 5 includes suggestions on alternate solutions and future work, such as testing alternative CNN architectures and encoding fingerprints in the clear using the client. We strongly suggest performing a survey comparing the performance of different encoding methods on cleartext fingerprints when paired with various privacy-preserving comparison algorithms, such as FingerCodes and GSHADE [33].

Additionally, usability could be greatly improved by investigating alternate key management solutions rather than using a physical token. Alternatively, integrating the token's functionality into devices commonly carried by potential users could be viable. For example, if the system is used to authenticate a car owner, a sufficiently advanced key fob could hold the data required by the system.

Finally, as applied FHE becomes more adopted and FHE-related libraries mature, it would be beneficial to continually compare viable candidates for implementing a full-scale version of the proposed system.

References

[1] Ilaria Chillotti et al. "TFHE: Fast Fully Homomorphic Encryption Over the Torus." In: *Journal of Cryptology* 33.1 (2020), pp. 34–91. ISSN: 0933-2790.

[2] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556.

[3] Randi Hindi. *Making CHATGPT encrypted end-to-end*. Apr. 2023. URL: https://www.zama.ai/post/chatgpt-privacy-with-homomorphic-encryption.

[4] K. Moriarty et al. *PKCS #1: RSA Cryptography Specifications Version 2.2*. RFC 8017. RFC Editor, Nov. 2016.

[5] Keith Martin. *Everyday Cryptography: Fundamental Principles and Applications*. Oxford University Press, June 2017. ISBN: 9780198788003. DOI: 10.1093/oso/9780198788003.001.0001.

[6] Ilaria Chillotti, Marc Joye, and Pascal Paillier. *Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks*. White Paper. Oct. 2020. URL: https://whitepaper.zama.ai/.

[7] Jon Callas et al. *OpenPGP Message Format*. RFC 2440. RFC Editor, Nov. 1998. URL: http://www.rfc-editor.org/rfc/rfc2440.txt.

[8] Jihyeon Ryu, Keunok Kim, and Dongho Won. "A Study on Partially Homomorphic Encryption". In: *2023 17th International Conference on Ubiquitous Information Management and Communication (IMCOM)*. 2023, pp. 1–4. DOI: 10.1109/IMCOM56909.2023.10035630.

[9] Pedro Silveira Pisa, Michel Abdalla, and Otto Carlos Muniz Bandeira Duarte. "Somewhat homomorphic encryption scheme for arithmetic operations on large integers". In: *2012 Global Information Infrastructure and Networking Symposium (GIIS)*. 2012, pp. 1–8. DOI: 10.1109/GIIS.2012.6466769.

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

[11] Craig Gentry. "Fully Homomorphic Encryption Using Ideal Lattices". In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC '09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178. ISBN: 9781605585062. DOI: 10.1145/1536414.1536440. URL: https://doi.org/10.1145/1536414.1536440.

[12] Jung Hee Cheon et al. "Homomorphic Encryption for Arithmetic of Approximate Numbers". In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 409–437. ISBN: 978-3-319-70694-8.

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

[14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *ACM Trans. Comput. Theory* 6.3 (July 2014). ISSN: 1942-3454. DOI: 10.1145/2633600.

[15] Tian Ye, Rajgopal Kannan, and Viktor K. Prasanna. "Accelerator Design and Performance Modeling for Homomorphic Encrypted CNN Inference". In: *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. 2020, pp. 1–7. DOI: 10.1109/HPEC43674.2020.9286219.

[16] Priyanka Sharma. "A Contemplate on MultiFactor Authentication". In: *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*. 2019, pp. 824–827.

[17] ISO/IEC 24745:2022. *Information security, cybersecurity and privacy protection — Biometric information protection*. Standard. Geneva, CH: International Organization for Standardization, Feb. 22.

[18] Danilo Valdes-Ramirez et al. "A Review of Fingerprint Feature Representations and Their Applications for Latent Fingerprint Identification: Trends and Evaluation". In: *IEEE Access* 7 (2019), pp. 48484–48499. DOI: 10.1109/ACCESS.2019.2909497.

[19] K Martin Sagayam et al. "Authentication of biometric system using fingerprint recognition with euclidean distance and neural network classifier". In: *Int. J. Innov. Technol. Explor. Eng* 8.4 (2019), pp. 766–771.

[20] Cambridge University Press. *Privacy*. In: *The Cambridge Business English Dictionary*. URL: https://dictionary.cambridge.org/dictionary/english/privacy.

[21] G. Dreyfus. "Neural Networks: An Overview". In: *Neural Networks: Methodology and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–83. ISBN: 978-3-540-28847-3. DOI: 10.1007/3-540-28847-3_1.

[22] Aanchal Vij and Kuldeep Singh Kaswan. "Prediction of Lung Cancer using Convolution Neural Networks". In: *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)*. 2023, pp. 737–741. DOI: 10.1109/AISC56616.2023.10085058.

[23] Sergio Saponara, Abdussalam Elhanashi, and Qinghe Zheng. "Recreating Fingerprint Images by Convolutional Neural Network Autoencoder Architecture". In: *IEEE Access* 9 (2021), pp. 147888–147899. DOI: 10.1109/ACCESS.2021.3124746.

[24] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[25] Yang Zhiqi. "Face recognition based on improved VGGNET convolutional neural network". In: *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. 2021, pp. 2530–2533. DOI: 10.1109/IAEAC50856.2021.9390856.

[26] Hossein Rajoli et al. "Triplet Loss-less Center Loss Sampling Strategies in Facial Expression Recognition Scenarios". In: *2023 57th Annual Conference on Information Sciences and Systems (CISS)*. 2023, pp. 1–6. DOI: 10.1109/CISS56502.2023.10089734.

[27] Oriol Vinyals et al. "Matching Networks for One Shot Learning". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

[28] Andrei Stoian et al. *Deep Neural Networks for Encrypted Inference with TFHE*. 2023. arXiv: 2302.10906 [cs.LG].

[29] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. "The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol". In: *Advances in Cryptology – EUROCRYPT 2019*. Cham: Springer International Publishing, 2019, pp. 129–158. ISBN: 978-3-030-17653-2.

[30] André Brasil Vieira Wyzykowski, Mauricio Pamplona Segundo, and Rubisley de Paula Lemes. *Level Three Synthetic Fingerprint Generation*. Licensed under the Attribution-NonCommercial-ShareAlike 4.0 International License (`https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode`). 2020. arXiv: `2002.03809`.

[31] Youssef Gahi, Mouhcine Guennoun, and Khalil El-Khatib. *A Secure Database System using Homomorphic Encryption Schemes*. 2015. arXiv: `1512.03498 [cs.CR]`.

[32] Mauro Barni et al. "Privacy-Preserving Fingercode Authentication". In: *Proceedings of the 12th ACM Workshop on Multimedia and Security*. MM&Sec '10. Roma, Italy: Association for Computing Machinery, 2010, pp. 231–240. ISBN: 9781450302869. DOI: `10.1145/1854229.1854270`.

[33] Julien Bringer et al. "GSHADE: Faster Privacy-Preserving Distance Computation and Biometric Identification". In: *Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security*. IH&MMSec '14. Salzburg, Austria: Association for Computing Machinery, 2014, pp. 187–198. ISBN: 9781450326476. DOI: `10.1145/2600918.2600922`.