Privacy preserving biometrics authentication in IoT devices using homomorphic encryption

Amir Dawd Seid
am4801se-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Qian Guo

Examiner: Thomas Johansson

May 23, 2024

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This thesis investigates the application of the Cheon-Kim-Kim-Song (CKKS) scheme in designing a privacy-preserving biometric authentication system in line with ISO 24745:2022 standard. Utilizing the Microsoft SEAL library the thesis integrates state-of-the-art facial recognition models to design client-to-cloud and cloud-to-client authentication systems. The solution is implemented in C++ and is tailored for IoT devices with constrained resources as a client while leveraging cloud computing for scalability and enhanced computational power. Given optimal security parameters, the implementation demonstrates that the system can authenticate users within five seconds.

Keywords: Homomorphic Encryption, Privacy-preserving, Biometric authentication, Microsoft SEAL, CKKS, DeepFace, OpenCV

Populärvetenskap

Föreställ dig att du närmar dig en dörr och den låser upp sig automatiskt utan att någonsin veta exakt vem du är? I examensarbetet har vi utforskat huruvida detta är genomförbart och vilka tekniska finesser möjliggör att en användare kan autentisera sig på ett säkert och integritetsbevarande sätt.

I takt med att vår värld blir alltmer digitaliserad ökar även behovet av säkerhetssystem som skyddar användares personliga information. Mitt examensarbete med titeln *Privacy preserving biometrics authentication in IoT devices using homomorphic encryption* har fokuserat på att utforska hur kryptering kan användas för att skapa en ny typ av autentiseringssystem som både är säkert och som respekterar användarnas integritet. Krypteringstekniken är revolutionerande på ett sätt att man faktiskt kan göra beräkningar medan data är i krypterad form. Låter för bra att vara sant... men hur är detta möjligt?

Tekniken heter homomorfisk kryptering och vars namn kommer från antika grekiskan. Det översätts till ”homo” samma och ”morf” form eller struktur. Alltså, datan bibehåller sin struktur även när den är i krypterad form. Homomorfisk kryptering är banbrytande teknik som möjliggör operationer på krypterade data utan att någonsin avkoda den. Detta innebär att känslig information såsom personnummer eller biometrisk information kan bearbetas utan att någon obehörig får tillgång till den faktiska datan. Den nya krypteringstekniken utvecklas hela tiden och öppnar dörren för diverse användarfall och appliceringsmöjligheter. Det finns olika implementationer av krypteringstekniken. CKKS är en sådan implementation och är optimerad för beräkningar på reella tal, vilket passar oss perfekt då vi arbetar med reella-tal när vi får ut biometrisk information ur ansiktsigenkännings modeller.

Studiens resultat visar att detta tillvägagångssätt inte bara höjer säkerhetsnivån utan också erbjuder en unik lösning på det klassiska dilemmat mellan integritet och tillgänglighet. Med CKKS kan vi utföra komplext autentiseringsarbete på ett sätt som tidigare inte varit möjligt och detta öppnar dörren för säker biometrisk autentisering.

Examensarbetet har tagit ett steg närmare att lösa några av de mest pressande digitala säkerhetsutmaningarna som vårt samhälle står inför idag. Det är en spännande tid för både teknik och integritetsbevarande åtgärder. Framtiden ser ljus ut för vidare utveckling och implementering av dessa avancerade kryptografiska metoder.

Acknowledgement

I want to express my gratitude to my supervisor Marie Åkesson, who listened to my initial proposal ideas and provided invaluable guidance and dedicated support throughout my thesis project. I am also thankful to Qian Guo for his supervision and continuous support. Last but certainly not least, I am grateful to my family for always being by my side and supporting me unconditionally.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Across many jurisdictions, privacy is a fundamental human right protected by data privacy regulations like the General Data Protection Regulation **GDPR**. GDPR encompasses a wide range of personal data for regulatory protection. That includes biometric data such as fingerprints, facial recognition, iris and retina, and other unique biological characteristics. Additionally, GDPR protects information that can directly or indirectly identify an individual, including but not limited to details like name, age and home address[1]. In today's data-driven world, ensuring user data privacy is not just a regulatory compliance matter but an absolute priority for organizations and businesses [2]. Therefore, privacy-enhancing technologies are needed to prevent sensitive data from being breached. A way to preserve the privacy of sensitive data is through encryption, which in turn requires cryptographic schemes [3].

1.1 Background

In 2019, a major data breach was discovered by security researchers in a biometric system. The system was mainly used by government firms, defence contractors, and banks. The belongings of the biometric system such as fingerprints, facial recognition information, personal details, and passwords of over 1 million people were found to be accessible in public databases. The biometric system was designed as an access control solution to secure facilities such as warehouses and office buildings. It was later revealed that user information was unprotected and mostly stored in unencrypted form. Furthermore, researchers stated that the scale of the breach was alarming due to its widely deployed service located in 1.5 million locations across the globe [4].

In early 2022, the ISO and IEC revised their standards and guidance on protecting biometric information, such as fingerprints and facial data, during the storage, transmission and processing phase. The ISO standard outlines guidelines and best practices to protect biometric information from unauthorized access, data breaches, or misuse. The standard proposes methods and tools for ensuring confidentiality, integrity and availability for secure user authentication. The standard is intended to be used by a wide range of organizations in the public and private sectors as well as technology vendors and service providers [5].

Homomorphism is a function that holds the same underlying condition given

an arbitrarily performed operation. The word *homomorphism* is of ancient-greek origin, *homo* (same) and *morph* (form) and roughly translates to *same-form*. The concept of privacy homomorphism was first coined by Rivest et al. in 1978, a year after the well-known crypto algorithm RSA was published [6]. Rivest et al. were aware of the RSA algorithm's multiplicative homomorphic property and suggested some potential applications. However, it took more than 30 years after its initial introduction to become applicable.

Homomorphic Encryption **HE** is an encryption method that supports arithmetic operations in the encrypted domain and is thus used in applications where privacy-preserving techniques are needed. Assuming m_1 and m_2 are plaintexts we want to encrypt, and $f(x)$ is the encryption function. An encryption scheme is said to be homomorphic if $f(m_1) * f(m_2) = f(m_1 * m_2)$. The first generation of HE was proposed in 2009 and supported additions and multiplications of encrypted data. Since then, HE has continuously improved, and its application has been extensively broadened.

Most asymmetric cryptosystems support one arithmetic operation homomorphically like RSA and Elgamal and are called a partial or somewhat homomorphic scheme. Furthermore, a cryptographic scheme that supports addition and multiplication is called fully homomorphic encryption **FHE**.

Privacy-preserving is a set of techniques and methods used to protect privacy when collecting, processing, analysing, or storing information of an individual or organizational entity. This thesis will mainly utilize homomorphic encryption to preserve data privacy while allowing for cloud-based computations.

1.2 Related work

In recent years, homomorphic encryption as a privacy-preserving technique for biometric authentication has been an active research area. However, most research papers have focused mainly on partial homomorphic encryption rather than fully homomorphic encryption. Upmanyu et.al proposed an efficient biometric verification system in an encrypted domain by utilizing the multiplicative homomorphic property of RSA. The system is therefore only limited to multiplication in the encrypted domain and addition is thus not possible [7].

Blaton et al proposed a biometric authentication protocol that employed a partially homomorphic encryption scheme, called the DGK encryption scheme. Their methodology centred around the computation of the Hamming distance between two iris feature vectors, resulting in remarkable performance outcomes with a computation time as low as 150 ms. Nonetheless, the protocol was constrained due to the additive-only nature of the employed encryption scheme [8].

After Gentry's introduction to the FHE scheme, the first example of biometric authentication based on the FHE scheme was published by Troncoso-Pastoriza et.al. The authors proposed a model that outsources the computation of encrypted facial templates using quasi-fully homomorphic encryption [9].

Later on, Boddeti presented a paper on how to execute secure face matching using the Fan Vercauteren FHE scheme and obtained practical results by packing the ciphertexts in a certain way [10]. With the advancements in fully homomorphic

encryption schemes, including the development of the TFHE library. Pradel and Mitchell's research improved the performance and practicality of FHE, making it more feasible for real-world applications. The proof-of-concept implementation in the paper used the TFHE library and showcased the underlying operations necessary for privacy-preserving biometric matching [11].

Jascha Kolberg presented in his PhD dissertation paper extensive research regarding Biometric Information Protection by utilising HE, building upon the foundational experiment by Drozdowski et al. (2019) [12]. Kolberg's study further advances the application of HE in biometric systems, particularly on facial identification leveraging FaceNet, ArcFace and the CKKS encryption scheme in comparison to other HE schemes, demonstrating the evolving field of secure biometric processing.

1.3 Objectives

The main objectives of the thesis are to conduct a research-oriented literature study and consequently implement a proof-of-concept end-to-end privacy-preserving access-control solution tailored for IoT devices. Furthermore, the thesis will evaluate obtained end-to-end results based on performance, and real-world applicability and then discuss the threats that pose to its security.

The security requirements for the proposed biometric authentication system in this thesis will be based on ISO:24745/2. Additionally, privacy requirements are presented to ensure that our solution is privacy-preserving.

1. Security requirements
    (a) Protect against unauthorized access (confidentiality).
    (b) Guard the consistency and accuracy of biometric data (integrity).
    (c) Deny access if required. (revocability).
    (d) Issue a new authorization access (renewability).
    (e) Accessible when resources are requested by an authorized entity (availability).
2. Privacy requirements
    (a) Biometric template information cannot be traced back to an individual (irreversibility)
    (b) Two biometric templates across different platforms deployed on the system shall not be linkable (unlinkability).
    (c) Prevent unauthorized access that threatens privacy leaks (confidentiality).

1.4 Contribution

Privacy-preserving authentication using HE is a research topic that has already been established. However, this thesis proposes a practical device-to-cloud and

cloud-to-device proof-of-concept implementation using the CKKS scheme. The thesis will evaluate the presented solution by studying the run-time performance, accuracy and most importantly how secure the system is. Consequently, the thesis proposes algorithms to perform computation on the cloud for secure privacy-preserving biometric authentication. Results presented in the thesis will also contribute to a better understanding and grounded implementation strategy when designing a privacy-preserving biometric system.

Chapter 2
Theoretical background

This chapter will introduce the basics and ground elements of abstract algebra and cryptology. Hence, the concepts are essential to understanding the contents of this paper. After introducing the core concepts, the theoretical background of homomorphic encryption, particularly the CKKS scheme, will be described in detail. Lastly, this chapter will conclude with the theoretical background of biometric authentication.

2.1 Preliminaries

2.1.1 Number theory

1. $\mathbb{N}$: Represents all natural numbers such as {0, 1, 2, 3, ...}.
2. $\mathbb{Z}$: Denotes whole numbers i.e. {..., -2, -1, 0, 1, 2, ...}.
3. $\mathbb{Q}$: Represents a number that can be expressed as the quotient of two integer numbers, assuming a and $b \in \mathbb{Z}$, $\mathbb{Q} \coloneqq \frac{a}{b}$, where $b \neq 0$.
4. $\mathbb{R}$: A number that can be expressed as a continuous decimal expansion.
5. $\mathbb{C}$: A number that can be expressed as $a + bi$, where a and $b \in \mathbb{R}$ and i is defined as $\sqrt{-1}$ that satisfies $i \times i = -1$.

2.1.2 Abstract algebra

Abstract algebra is a field study of algebraic structures, in which for instance groups, rings, fields, and lattices are included. In this section sets, groups, and rings will be described briefly.

Set

A Set S is a collection of objects called elements or members and is noted using a curly bracket. Assuming $S = \{1, 2, 3, 4, ...\}$, properties of S can be noted as $S = \{x \in \mathbb{Z} \mid 1 \text{ divides } x\}$ and is read as S consists of integer elements such that each element is divisible by one.

Assuming T is a set and each element in T is an element in S, we denote that as $T \subset S$, meaning T is contained in or is a subset of S [13].

Map

A map f is a rule that assigns each element of T to an element in S. A map f is denoted as, $f : T \to S$, whereas T is the domain and S codomain. For a specific element x in T, a value that gets mapped to an element in S is denoted as $x \mapsto f(x)$, where f is the rule or function and $f(x)$ is the resulting output.

Group

A group G is a map of sets that operates two elements in G and results in the third element within G, meaning $* : G * G \to G$.

For a set to be defined as a group, it needs to fulfil the following requirements:

- G is closed, $a * b \in G$, $\forall a, b \in G$.
- G is associative, meaning $a * (b * c) = (a * b) * c$. $\forall\ a, b, c \in G$.
- Identity element exists in G i.e. $\exists\ 1\ ,\ 1 \in G$, $1 * a = a * 1 = a$. $\forall a, a \in G$.
- There is an inverse in G, $a \in G$, $\exists a^{-1} \in G$ such that $a * a^{-1} = 1$

A group that has commutative property meaning, $a * b = b * a$ is called an *Abelian group*, $\forall a, b \in G$. If we assume $G = \mathbb{Z}$, an integer group that has additive operation is denoted as $(\mathbb{Z}, +)$ [13].

Ring

A ring R is a set that is equipped with two binary operations referred to as addition $+$ and multiplication $*$. A ring has to satisfy the following conditions.

- R is an abelian group under additive operation, meaning $a + b = b + a$, $\forall a, b, \in R$ and the identity element is 0, where $0 \in R$ and satisfies $a + 0 = 0 + a = a$, $\forall a \in R$.
- R is associative under multiplication, $a * (b * c) = (a * b) * c$, $\forall a, b, c \in R$ and there exists an identity element under multiplication, $1 \in R, 1 * a = a * 1 = a$ $\forall a \in R$.
- Distributive property applies under addition and multiplication, meaning $a * (b + c) = a * b + a * c \forall$ a, and $a, b, c \in R$.

Homomorphism

Homomorphism is a map or a function that preserves the same operation and structure between groups, rings, or vector spaces. Assuming we have two groups i.e. G and H, and f as a function, $f : G \to H$. A map is said to be homomorphic if it fulfils the following equivalence: $f(x * y) = f(x) * f(y)$ $\forall x, y \in G$. Additionally, the function must preserve the same underlying structure under addition and multiplication for rings, furthermore vector addition and scalar multiplication for vector spaces [14].

Embedding

An embedding is a function that maps elements from one mathematical structure into another while preserving structural properties. Assuming G and H are groups, then an embedding $\sigma : G \to H$ would ensure that for any $g_1, g_2 \in G$, then the relation $\sigma(g_1 * g_2) = \sigma(g_1) * \sigma(g_2)$ still holds in H and the homomorphic property is still preserved. Furthermore, if σ is bijective then there is an isomorphism between G and H.

Polynomial ring

A polynomial in one variable over a ring is defined as $f(x) = \sum_{i=0}^{n} a_i x^i$, where $a_n \neq 0$ and each coefficient $a_i \in R$ and the degree of $f(x)$ is determined by n. The set of all such polynomials forms a polynomial ring, denoted as $R[X]$. A polynomial is said to be *monic* if its leading coefficient is 1. For instance, the polynomial ring with integer coefficients is denoted as $\mathbb{Z}[X]$ [14].

Cyclotomic polynomial

A cyclotomic polynomial is a monic irreducible polynomial over $\mathbb{Q}$ with integer coefficients whose roots are the primitive nth roots of unity. The complex nth roots of unity are defined as the solution to $x^n = 1$, represented as $e^{2i\pi\frac{k}{n}}$ for $k = 0, 1, 2, ..., n-1$. The n-th cyclotomic polynomial is defined as follows:

$$\Phi_n(x) = \prod_{\substack{1 \leq k < n \\ gcd(k,n)=1}} (x - e^{2i\pi\frac{k}{n}}) \tag{2.1}$$

Where k and n are relatively prime numbers [15].

2.2 Cryptography

Cryptography is the study and the techniques required for two parties to communicate securely in the presence of a third party often referred to as an adversary. The importance of protecting information has been a priority since ancient times. The Scytale was a tool used by ancient Spartans to transmit information concerning their military campaigns. Another notable method is the Ceasar Cipher, one of the earliest cryptographical schemes that can be dated back to 100 BC and was an encryption scheme that used the alphabet system, wherein each letter was shifted to the right or left N times to produce a ciphertext and is referred as shift-cipher [16].

Building upon these foundational principles of safeguarding information, modern-day cryptographic techniques have evolved to be significantly more advanced and secure. There are two main types of cryptographic schemes, symmetric and asymmetric key-based cryptography.

2.2.1 Symmetric cryptography

In symmetric key cryptography, the same key is used to encrypt and decrypt a message. This means that the sender and the receiver must both possess the same key to communicate securely. Examples of symmetric key algorithms include AES and DES [17].

Symmetric key algorithms are prevalent due to their speed and memory efficiency and are well-suited for bulk data encryption. However, one major limitation of symmetric key cryptography is the need for the sender and receiver to share a secret key. This can be problematic when the parties do not have initiated secure communication channels.

2.2.2 Asymmetric cryptography

Asymmetric key cryptography also referred to as public key cryptography, uses a pair of keys. The public key is used to encrypt a message, while the private or secret key is used to decrypt the message. This allows for secure communication without the need for the sender and receiver to share the same secret key. Examples of asymmetric key algorithms include RSA, ElGamal and ECC.

Asymmetric key algorithms are slower and less efficient than symmetric ones, though they offer several advantages. One major benefit is the public key can be shared openly, while the private key remains secret. This enables secure communication without the need for a pre-existing relationship between the sender and receiver. Asymmetric key algorithms are used for digital signatures, which are utilised to authenticate the identity of the sender and the integrity of the message.

2.3 Notion of security

The notion of cryptographic schemes refers to the mathematical foundations that underlie the various algorithms and protocols used in cryptography. These schemes are designed to be computationally difficult, requiring significant time and resources to solve. This makes it difficult for adversaries to break the encryption and access the protected information [17].

2.3.1 Factoring problem

One well-known cryptographic challenge is the Integer Factoring Problem. This involves finding the prime factors of a large composite integer. The security of many modern cryptographic algorithms in public-key cryptography is based on the assumption that factoring large composite integers is computationally infeasible. This computational difficulty ensures that an adversary cannot easily determine the original message from its encrypted version. For instance, the RSA scheme is based on this problem.

2.3.2 Discrete Logarithm Problem

Another critical challenge in cryptography is the Discrete Logarithm Problem. This involves determining an integer x that satisfies the equation $y = g^x \mod p$,

where g and p are known constants, and y is a given value. The security of cryptographic algorithms such as the Diffie-Hellman key exchange and the ElGamal encryption system relies on the assumption that solving the Discrete Logarithm Problem is computationally infeasible for large values of p.

2.3.3 Lattice Problem

A third example within cryptographic foundations is the Ring Learning with Errors **RLWE** problem. This problem is not about finding an integer x but rather involves solving for polynomial rings over finite fields that satisfy an equation of the form $a(x) \cdot s(x) + e(x) = b(x) \mod q$, where $a(x), e(x), b(x)$, and q are known, and $s(x)$ is the secret polynomial to be determined. The security of various post-quantum cryptographic algorithms leverages the assumption that solving the RLWE problem is computationally infeasible. RLWE-based schemes are fundamental in constructing modern cryptographic algorithms and protocols, especially if aiming for security against quantum computer-based attacks. RLWE is based on a hard lattice problem and is viewed as post-quantum secure.

2.4 Fully Homomorphic Encryption

Fully homomorphic encryption is a public key-based encryption scheme that encompasses arbitrary homomorphic properties. In 2009 Craig Gentry introduced the first FHE scheme in his Ph.D. thesis titled *A FULLY HOMOMORPHIC ENCRYPTION SCHEME* [18]. Before Gentry's breakthrough, HE schemes could either perform addition or multiplication, but not both operations simultaneously. FHE schemes are defined through the following key generation, encryption, decryption and evaluation circuit algorithms.

- **KeyGen(λ)**: Probabilistic algorithm that takes in a parameter λ and outputs sk and pk.
- **Encrypt(m, pk)**: Probabilistic algorithm that takes in a message m, the public key pk and results in a ciphertext c.
- **Decrypt(c, sk)**: Deterministic algorithm that takes in c, sk and produces m.
- **Evaluate(f, c_1 ... c_n; pk)**: Takes in pk, n amounts of ciphertexts $c_1 ... c_n$ and a computational circuit f.

2.4.1 First Fully Homomorphic scheme

The first FHE scheme starts by constructing a somewhat-HE scheme, which is limited to low-degree polynomials. The limitation is mainly due to some noise added to the ciphertext for security purposes and the noise grows after each computation which results in a larger overhead and affects the performance. Furthermore, the scheme has limited multiplicative depth, meaning there is a limitation on how much computation one can perform and when the noise reaches a certain threshold, the decryption algorithm will not yield the correct result.

To tackle the issue of increasing noise overhead, Gentry proposed a method called bootstrapping. Bootstrapping is a technique used to refresh the ciphertext by encrypting the secret key and running the decryption evaluation homomorphically. This procedure is performed iteratively and recursively to reduce the noise overhead of the ciphertext, thus allowing unlimited computation on encrypted data.

Although Gentry's FHE scheme was groundbreaking, it took about 30 mins to perform two-bit operations and therefore was impractical to be applied in real-world applications [18].

There have been many refinements and improvements after Gentry's first introduction to FHE. Researchers have proposed improved FHE schemes that are much more efficient, and some of them are now practical enough for specific real-world applications. Some of them are NTRU, BGV, BFV TFHE and the CKKS scheme. In the upcoming section, we will describe the CKKS scheme extensively as it is our underlying choice for implementation.

2.5 CKKS encryption scheme

Cheon-Kim-Kim-Song **CKKS** is an encryption scheme that offers arithmetic operations on encrypted messages. Although other homomorphic encryption schemes can only operate on integers, CKKS allows the process of floating point numbers and supports approximated operations such as addition, multiplication and scaling. The scheme operates by encoding and scaling up input floating-point numbers using a pre-chosen scaling factor parameter Δ to transform them into integers. The resulting output of the encrypted message is then scaled down using the same scaling factor Δ^{-1} to produce an approximated plaintext [19].

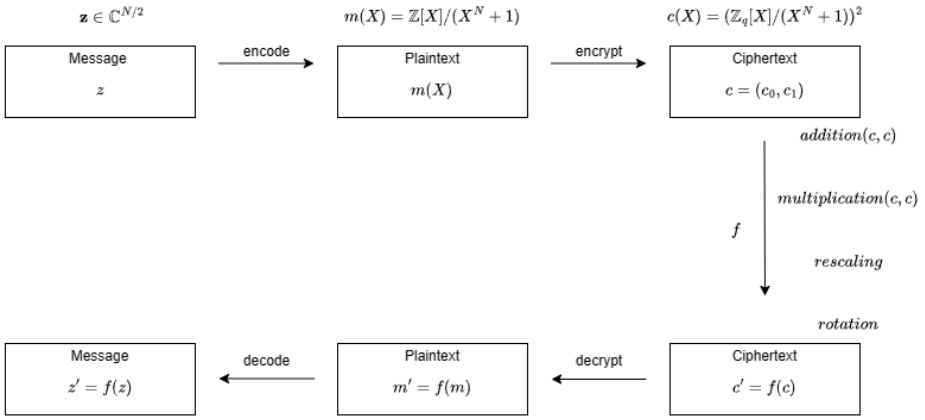


Figure 2.1: Overview of the CKKS scheme.

CKKS suits our application because biometric templates are often represented as floating point vectors.

2.5.1 Simple encoding

CKKS uses polynomial rings with complex coefficients to encode input vector messages, called plaintext. The vector consists of floating-point numbers and is mapped into a cyclotomic polynomial ring that facilitates arithmetic operations on encrypted data. The scheme takes in input vector $\mathbf{z}$ of size $N/2$, denoted as $\mathbf{z} = [z_1, z_2, ..., z_{N/2}] \in \mathbb{C}^{N/2}$, where N is a power-of-two integer.

The input vector $\mathbf{z} \in \mathbb{C}^{N/2}$ is then mapped into a cyclotomic polynomial ring $m(X) = \mathbb{C}[X]/\Phi_M(X) = \mathbb{C}[X]/(X^N+1)$, where N in this case the degree modulus of $m(X)$.

The encoding procedure takes the input vector message $\mathbf{z}$ and the scaling factor $\alpha > 1$. The choice of α influences the precision and the number of homomorphic operations that can be performed due to the growth of noise within the ciphertext.

Embedding

An embedding is a mathematical structure contained within another structure that still preserves its algebraic properties. We will start by embedding the polynomial $m(x)$ into an input vector for a better understanding. In this section we assume, $m(x) \in \mathbb{C}[X]/(X^N + 1)$.

$$\sigma : \mathbb{C}[X]/(X^N + 1) \to \mathbb{C}^N \tag{2.2}$$

The embedding process is done by evaluating the encoded polynomial $m(X)$ on values that are the roots of unity for $\Phi_M(X) = X^N+1$, where $M = 2N$ and the N-th root is given such as $e^{2i\pi\frac{K}{M}}$, where $K = \{1, 3, ..., 2N-1\}$ is an odd exponent that ensures that all N distinct roots of unity that are needed for the canonical embedding are used.

$$\sigma(m) = (m(e^{2i\pi\frac{K_0}{M}}), m(e^{2i\pi\frac{K_1}{M}}), ..., m(e^{2i\pi\frac{K_n}{M}}) = [z_1, z_2, ..., z_n] = \mathbf{z} \tag{2.3}$$

The embedding process is isomorphic, meaning the input vector will be mapped to a unique polynomial and the polynomial will be mapped to a distinct input vector.

Inverse embedding

To map the input vector $\mathbf{z} \in \mathbb{C}^N$ into cyclotomic polynomial $m(X)$ we use the following inverse embedding:

$$\sigma^{-1} : \mathbb{C}^N \to \mathbb{C}[X]/(X^N + 1) \tag{2.4}$$

To encode the input vector into a polynomial we need to find a polynomial $m(X)$ such that $m(X) = \sum_{i=0}^{N-1} \alpha_i X^i \in \mathbb{C}[X]/(X^N + 1)$, given the input vector $\mathbf{z} \in \mathbb{C}[\mathbb{X}]$ and that satisfies $\sigma(m) = \mathbf{z}$. This entails that we need to find the coefficients α_i and thus solve the following equation.

$$\sum_{k=0}^{N-1} \alpha_k (e^{2i\pi\frac{2n-1}{M}})^k = z_n, n = \{1, .., N\} \text{ and } M = 2N. \tag{2.5}$$

This results in a linear equation that can be solved as $A\alpha = \mathbf{z}$, A representing the Vandermonde matrix of $(e^{2i\pi\frac{2n-1}{M}})_{n=1,...,N}$. We define $\zeta = (e^{\frac{i\pi}{n}})$ and α being the polynomial coefficient and $\mathbf{z}$ the input vector to be encoded.

$$A = \begin{bmatrix} \zeta^0 & \zeta^1 & \zeta^2 & \cdots & \zeta^n \\ (\zeta^3)^0 & (\zeta^3)^1 & (\zeta^3)^2 & \cdots & (\zeta^3)^n \\ 1 & (\zeta^5)^1 & (\zeta^5)^2 & \cdots & (\zeta^5)^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (\zeta^{2n-1})^1 & (\zeta^{2n-1})^2 & \cdots & (\zeta^{2n-1})^n \end{bmatrix} \tag{2.6}$$

To obtain the coefficients we solve the following equation.

$$\alpha = A^{-1}\mathbf{z}^{\mathbf{T}} \tag{2.7}$$

In this section we mapped complex input vectors $\mathbf{z} \in \mathbb{C}^{N/2}$ into a cyclotomic polynomial ring $\mathbb{C}[X]/(X^N + 1)$. However, we need to do more steps to map $\mathbf{z} \in \mathbb{C}^{N/2}$ into $\mathbb{Z}[X]/(X^N+1)$. In the upcoming steps we will denote $Z[X]/(X^N+1)$ as $\mathcal{R}$.

2.5.2 Full encoding

Full encoding ensures that the encoded polynomials have integer coefficients, which is necessary for the CKKS scheme to leverage the properties of polynomial integer rings. Therefore, we will map the input values of $\mathbf{z} \in \mathbb{C}^{N/2}$ to $\mathcal{R}$.

Inverse projection

Assuming $\mathbb{H}$ satisfying the following condition $\mathbb{H} = \{(z_j)_{j\in\mathbb{Z}_M^*} : z_j \in \mathbb{C}, z_j = \overline{z}_{-j}, \forall j \in \mathbb{Z}_M^*\}$. The set $\mathbb{H}$ defines the conjugate symmetry when evaluating polynomials at the roots of unity. Then we define the natural projection as $\pi : \mathbb{H} \to \mathbb{C}^{N/2}$.

The inverse projection is then $\pi^{-1} : \mathbb{C}^{N/2} \to \mathbb{H}$, such that $[z_1, ..., z_{\frac{n}{2}}] \to [z_1, ..., \overline{z}_n]$.

Scaling

To adjust the precision and scale of the plaintext coefficients, the vector $\mathbf{z}$ is scaled by a factor $\Delta \in \mathbb{Z}^+$.

$$\mathbf{z}' = \Delta \cdot \hat{\mathbf{z}} \tag{2.8}$$

where $\mathbf{z}' \in \Delta \cdot \mathbb{C}^{N/2}$.

Lattice projection

Direct projection $\sigma : \mathcal{R} = \mathbb{Z}[X]/(X^N + 1) \to \sigma(\mathcal{R}) \subseteq \mathbb{H}$ is not possible, as not every element of $\mathbb{H}$ is in $\sigma(\mathcal{R})$. Although σ defines an isomorphism from $\mathcal{R}$ to $\sigma(\mathcal{R})$, it is not surjective onto $\mathbb{H}$.

However, we can project the vector $\mathbf{z}' \in V$ onto the lattice of the ring $\mathcal{R}$.

The isomorphism V is given by

$$V : \mathcal{L}(\mathbb{Z}[X]/(X^N + 1)) \to \mathbb{Z}[X]/(X^N + 1) \tag{2.9}$$

and is expressed in terms of the orthogonal lattice basis. The vector $\mathbf{z}'$ is denoted as $\mathbf{z}' = [z'_1, \ldots, z'_n]$ and is represented as a linear combination of the basis vectors A_i, with each A_i being the i-th column of the Vandermonde matrix A.

The coordinate vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ is obtained by computing the dot product of $\mathbf{z}'$ and A_i, shown below.

$$\mathbf{a} = \left(\frac{(\mathbf{z}', A_1)}{(A_1, A_1)}, \ldots, \frac{(\mathbf{z}', A_n)}{(A_n, A_n)} \right) \tag{2.10}$$

The projection is completed by rounding each coordinate value in $\mathbf{a}$ to the nearest integer.

$$\mathbf{a} \to \lceil \mathbf{a} \rfloor \tag{2.11}$$

Decoding

For decoding, we apply the inverse of the encoding process.

$$\mathbf{z} = \pi \circ \sigma \left(\frac{m(X)}{\Delta} \right) \tag{2.12}$$

where the canonical embedding σ is applied to $m(X)$ divided by the scaling factor. The natural projection π reduces the resulting vector to $\mathbb{C}^{N/2}$, giving us the approximated input vector.

See Appendix A.1 for the full encoding and decoding procedure. A.1

2.5.3 Key generation

Before delving deeper into the key generation process, let's introduce the following distributions. First, consider the discrete Gaussian distribution $\mathcal{D}G(\sigma^2)$, where the standard deviation σ is positive. We sample a vector in $\mathbb{Z}^N$, where each coefficient is independently chosen from the discrete Gaussian distribution based on the variance σ^2. Another distribution is the hamming weight sampling distribution $\mathcal{HWT}(h)$ that is represented as the set of vectors in $\{-1, 0, 1\}^N$, whose hamming weight i.e. non-zero values are exactly h. Furthermore, we have the distribution $\mathcal{ZO}(\rho)$, where $\rho \in \mathbb{R}$ and $0 \leq \rho \leq 1$. The distribution picks each entry in the vector $\{-1, 0, 1\}^N$, for picking $\{\pm 1\}$ the probability is $\rho/2$ and $1 - \rho$ for $\{0\}$.

The cyclotomic polynomial ring is $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, as denoted in previous section. Furthermore, $\mathcal{R}_{q_L} = \mathbb{Z}_{qL}[X]/(X^N + 1)$.

Initialization

Based on a security parameter λ, we pick M, where M $= 2^k$ and is function of $M = M(\lambda, q_L)$, pick $h = h(\lambda, q_L)$ and $P = P(\lambda, q_L)$ where h and P are integers. Choose $\sigma = \sigma(\lambda, q_L)$, where $\sigma \in \mathbb{R}$.

Secret key

We sample $s \leftarrow \mathcal{HWT}(h)$. The secret key sk is then sampled as $sk \leftarrow (1, s)$.

Public key

Sample $a \leftarrow \mathcal{R}_{q_L}$, $e \leftarrow \mathcal{DG}(\sigma^2)$ and $b \leftarrow -a * s + e \bmod q_L$. Then the public key pk is sampled as $pk \leftarrow (b, a) \in \mathcal{R}^2_{q_L}$.

Evaluation key

Sample $a' \leftarrow \mathcal{R}_{Pq_L}$, $e' \leftarrow \mathcal{DG}(\sigma^2)$ and $b' \leftarrow -a' * s + e' + Ps^2 \bmod (Pq_L)$. Then the evaluation key evk is sampled as $evk \leftarrow (b', a') \in \mathcal{R}^2_{Pq_L}$. P is the same value as chosen in the initialisation phase.

2.5.4 Encryption

The encryption algorithm encrypts the message m with the public key pk, a value v sampled from $v \leftarrow \mathcal{ZO}(0.5)$ and error values $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$.

The ciphertext $c = Enc_{pk}(m) = v * pk + (m + e_0, e_1) \pmod{q_L}$

2.5.5 Arithmetic operation

Considering the following ciphertexts $c_1, c_2 \in \mathcal{R}^2_{q_\ell}$, addition is then simply done by summing the ciphertexts under mod q_ℓ. However, both ciphertexts must have the same scaling factor α with the following assumption: $\alpha < \ell < L$.

$$add(c_1, c_2) = c_1 + c_2 \mod q_\ell \tag{2.13}$$

For multiplication, we use the evaluation key evk. The evaluation or relinearization key as it is also referred to is mainly used to minimise the growth of noise during multiplication. Also here, we assume the scaling factor is the same for c_1 and c_2. Furthermore, we consider that $c_1 = (b_1, a_1)$ and $c_2 = (b_2, a_2)$. If we multiply c_1 and c_2 component-wise, we get the following.

$$x_0 = b_1 b_2 \tag{2.14}$$
$$x_1 = b_1 a_2 + a_1 b_2 \tag{2.15}$$
$$x_2 = a_1 a_2 \tag{2.16}$$

$$mult_{evk}(c_1, c_2, evk) = (x_0, x_1) + \lfloor P^{-1} * x_2 * evk \rceil \mod q_\ell \tag{2.17}$$

P^{-1} is the multiplicative inverse of P introduced in 2.5.3 and is used to adjust the scaling of the result after multiplication.

2.5.6 Rescaling

The rescaling procedure decreases the ciphertext by one level. For $c \in \mathcal{R}^2_{q_\ell}$.

$$RS_{\ell \to \ell'}(c) = \left\lfloor c \cdot \frac{q_{\ell'}}{q_\ell} \right\rceil \mod q'_\ell \tag{2.18}$$

where $\ell' = \ell - 1$

2.5.7 Decryption

The ciphertext c, represented as as $c = (b, a)$ is decrypted using the secret key $sk \leftarrow (1, s)$. The decryption process is as follows:

$$dec_{sk}(c) = b + as \mod q_\ell \tag{2.19}$$

2.6 Biometric authentication

Biometrics encompasses data that uniquely identify an individual based on their physical traits and behavioural characteristics, such as fingerprints, facial patterns, iris structure, and voice [5]. Biometric authentication, involves using these biometric credentials for identity verification in various security applications.

Biometric systems are built upon three main processes: enrollment, identification and verification.

2.6.1 Enrollment

Enrollment is the first process an individual presents biometric data such as facial patterns. The system extracts specific features from the raw biometric data to create a unique biometric template using recognition models.

Recognition models

This thesis will utilise four recognition models as our underlying biometric facial recognition models, GhostFaceNets, ArcFace, SFace and FaceNet. The recognition model's accuracy is based on the LFW dataset.

- **GhostFaceNets**, published in 2023 is a model tailored for devices with limited memory and computational resources with state-of-the-art accuracy of 99.8667% [20].
- **ArcFace** is a model built upon deep neural networks through an angular margin loss, significantly improving face verification and identification performance. The model was published in 2018 and has 99.83% accuracy [21].
- **SFace** is a privacy-friendly recognition model trained on synthetic data. The model was made public in 2022 and scored 99.13% [22].
- **FaceNet** is a model from 2015 that is trained on mapping facial images to their respective representations in Euclidean space. The model reported an accuracy of 99.63% [23].

Feature extraction

Feature extraction in recognition models involves a deep learning model, used to identify and isolate key attributes from raw facial images. This step transforms high-dimensional image data into a one-dimensional vector space [12].

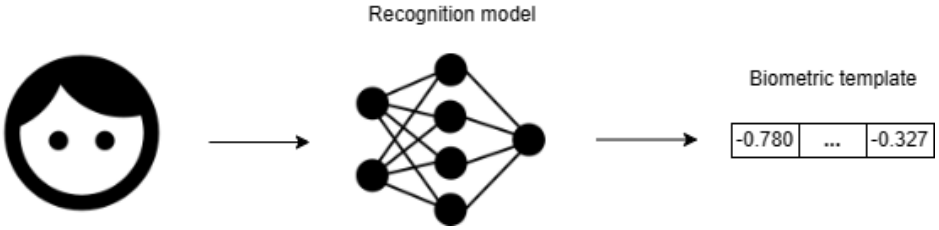


Figure 2.2: Feature extraction.

Feature normalization

Feature normalization is a pre-processing step that transforms the biometric template vector into a normalized vector that maintains its original direction but is scaled to lie on the unit sphere. This step helps standardize the range of feature values.

$$\hat{X} = \frac{X}{\|X\|} \tag{2.20}$$

where $\|X\| = \sqrt{X_0^2 + X_1^2 + ... + X_n^2}$ and is defined as the euclidean norm of X.

2.6.2 Identification and verification

The identification process searches the biometric features collected from a user against a database of enrolled biometric templates. This step determines if the individual's features match any of the profiles in the database. The system identifies the individual directly or lists potential matches if multiple individuals are found [5]. The verification process challenges the identity claim by providing raw data to match a specific template enrolled in the system. Consequently, the outcome of both identification and verification processes in biometric systems is based on the degree of similarity between the incoming biometric data and the stored templates.

Cosine similarity

Cosine similarity measures the cosine of the angle between two vectors in a multidimensional space, a metric used to determine how similar the vectors are regardless of their length or magnitude. In biometric systems, cosine similarity is used to compare normalized vectors of extracted features. The outcome of cosine similarity ranges between $[-1, 1]$, where -1 indicates that the vectors are diametrically opposite of each other, 0 orthogonal thus no correlation and 1 means they have the same orientation therefore full correlation [23].

$$cos(\theta) = \frac{X \cdot Y}{\|X\|\|Y\|} \tag{2.21}$$

where $\|X\|$ and $\|Y\|$and are the euclidean norms of X respective Y. Alternatively, the cosine function can be reformulated as:

$$cos(\theta) = \hat{X} \cdot \hat{Y} \tag{2.22}$$

When X and Y are normalised.

Euclidean distance

Euclidean distance measures the distance between two points in a multidimensional space. The Euclidean distance is often used to assess the dissimilarity between feature vectors representing different individuals.

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2} \tag{2.23}$$

City Block distance

The City Block distance, also referred to as the Manhattan distance measures the sum of the absolute differences between two points and their Cartesian coordinates. The Manhattan distance is useful for comparing features that require aggregation of linear distances rather than geometric distances.

$$d_{\text{Manhattan}}(X,Y) = \sum_{i=1}^{n}|X_i - Y_i| \tag{2.24}$$

2.7 Biometric evaluation

Biometric evaluation is the process that assesses the performance of a biometric system. It involves verifying how well the system can recognize and confirm a user by studying the key metrics described below.

2.7.1 False Acceptance Rate

False Acceptance Rate **FAR** quantifies the probability that the system will incorrectly authorize an access attempt by an unauthorized user.

$$\text{FAR} = \frac{\text{Total False Acceptance}}{\text{Total False Attempts}} \tag{2.25}$$

2.7.2 False Rejection Rate

False Rejection Rate **FRR** is a measure of the likelihood that the system will incorrectly reject an access attempt by an authorized user.

$$\text{FRR} = \frac{\text{Number of false rejections}}{\text{Total number of genuine authentication attempts}} \tag{2.26}$$

2.7.3 ROC curve

The Receiver Operating Characteristic **ROC** curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its threshold is varied. The curve plots the True Positive Rate **TPR** against the False Positive Rate **FPR**

at various threshold settings. The ROC curve is used to illustrate the trade-off between the system's sensitivity TPR and specificity $1 - FPR$, which provides a comprehensive overview of the system's performance across different thresholds.

$$\text{TPR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.27}$$

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \tag{2.28}$$

Chapter 3
Approach

This section will start by describing the architectural overview of the overall modules in our system, inspired by the ISO/IEC 24745 standard on biometric information protection. Moreover, this section will showcase the design and structural interplay between different modules in our biometric system to preserve data privacy security.

3.1 Architectural overview

The solution in our thesis is largely inspired by the Model K store distributed/-compare distributed framework, as described in the ISO/IEC 24745:2022 standard. This model serves as the foundation for our system's design, ensuring that the storage and comparison of biometric data are distributed across client and server entities to maintain the privacy of user information. By leveraging the CKKS scheme, our architecture enables secure computations on encrypted data by ensuring that sensitive biometric information remains encrypted throughout the process. The decision is based on the cosine similarity and Euclidean distance, computed homomorphically on the cloud side. See Figure 3.1 for a detailed illustration.

3.2 Dataset and tools

3.2.1 LFW

We utilize the Labeled Faces in the Wild **LFW** dataset to evaluate our biometric authentication system. This dataset provides a collection of facial photographs of celebrities and contains 13233 images of 5749 unique individuals and is designed for facial verification. The photos were collected from the web and represent a broad range of real-world conditions with variations in lighting, pose and background [25]. The LFW dataset is thus a suitable resource for testing the robustness of our privacy-preserving biometric authentication system under varied and realistic conditions.

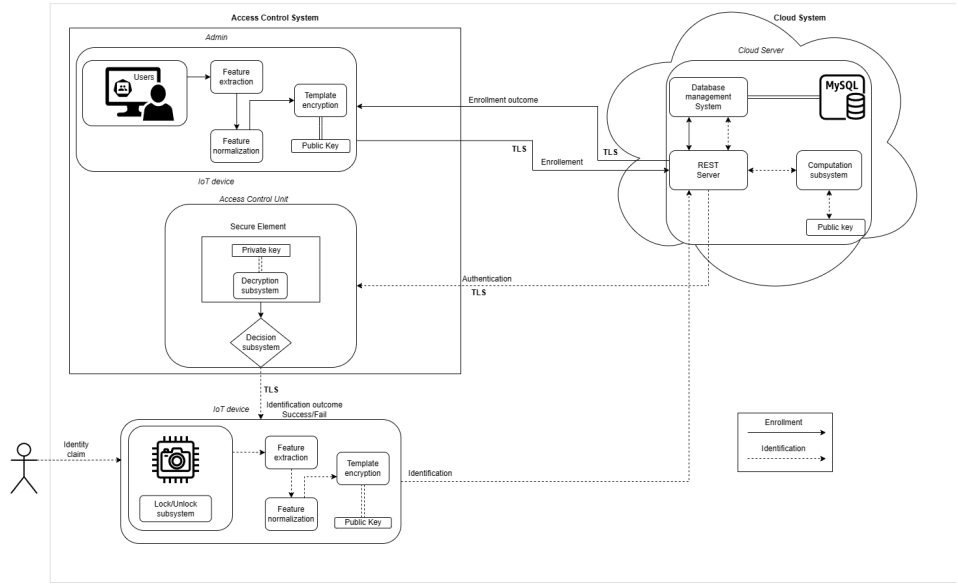


Figure 3.1: Architectural overview.

3.2.2 SEAL

Simple Encrypted Arithmetic Library **SEAL** is an open-source, cross-platform homomorphic encryption library in modern standard C++. The library is easy to compile and integrate into different platforms and applications and supports the BGV, CKKS and BFV homomorphic schemes [26].

SEAL will be used as an underlying library in our project for both key generation and the execution of homomorphic computations required by the CKKS scheme.

3.2.3 Blake3

Blake3 is a cryptographic hash function that builds on Blake2 and the enhancements of the Chacha stream cipher. Blake3 was announced in 2020 and is designed to be secure and fast, outperforming SHA-3 and Blake2 in terms of speed. Consequently, we have chosen Blake3 as the hash function to hash the static UUID in our solution. The library is implemented in C and Rust and is licensed under CC0 and Apache Licence [27].

3.2.4 OpenCV

Open Source Computer Vision Library **OpenCV** is a real-time computer vision library developed by Intel. The library is under an open-source BSD license and supports Windows, Linux, Mac OS, and Android. OpenCV is written in C++ and has Python, Java, and MATLAB interfaces. The library supports a wide range

of applications in machine vision, including facial recognition technology, gesture recognition, and motion tracking [28].

The IoT device in our biometric authentication system utilizes OpenCV to process real-time video, captured through its onboard camera that mainly focuses on detecting faces and then triggering other subprocesses.

3.2.5 DeepFace

DeepFace is an open-source lightweight facial recognition library wrapped in state-of-the-art models such as GhostFaceNet, Google FaceNet, Facebook DeepFace, SFace and ArcFace [29].

3.3 Key generation and distribution

The key generation and distribution is an integral part of our solution and handles the creation, storage and distribution of the private, public, relinearisation and galois keys of the CKKS scheme. The key generation process is therefore performed upon the initialisation phase of the system by the Access Control Unit **ACU**. The ACU stores the private key within its secure element, while CKKS parameters such as scaling factor, context and public keys are sent to the IoT device and Admin over secure TLS communication. Furthermore, relinearisation and galois keys as well as the CKKS parameters are distributed to the Cloud System to enable computations on encrypted data.

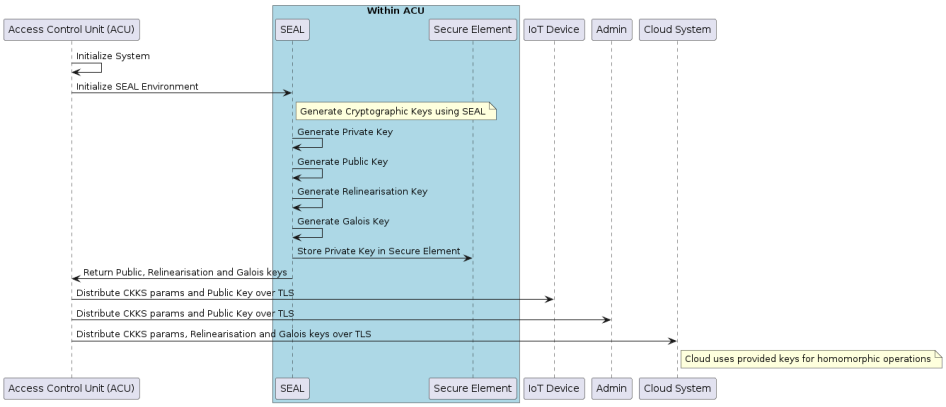


Figure 3.2: Key generation and distribution.

3.4 Admin

The Admin has a pivotal role in our solution and ensures that the enrollment process adheres to the guidelines set by ISO/IEC 24745, safeguarding the integrity and confidentiality of biometric data from the initial stages of user registration. The Admin also manages the revocation of user access to protect against unauthorized

access and ensures that resources are only accessible to authorised users. Moreover, the renewability of credentials is also handled by the admin for maintaining system security and user access legitimacy over time.

We have proposed a Unique User Identifier **UUID** to identify users with distinct letters or numbers that distinguish them from others. The identifier can for instance be a card serial number.

3.4.1 Enrollment

The enrollment process in our biometric authentication system starts with the user providing a UUID to the Admin. This identifier is then securely hashed using Blake3, enhancing user anonymity. Following this, the user's facial image is captured via OpenCV and forwarded to DeepFace for feature extraction. The extracted features are normalized by the Admin and encrypted using the SEAL library with a public key.

The encrypted data and the hashed identifier are transmitted securely over TLS to the Cloud System, which forwards it to the Database for storage. The database confirms the outcome of the storage operation back to the Cloud System, which then informs the Admin of the success or failure of the process.

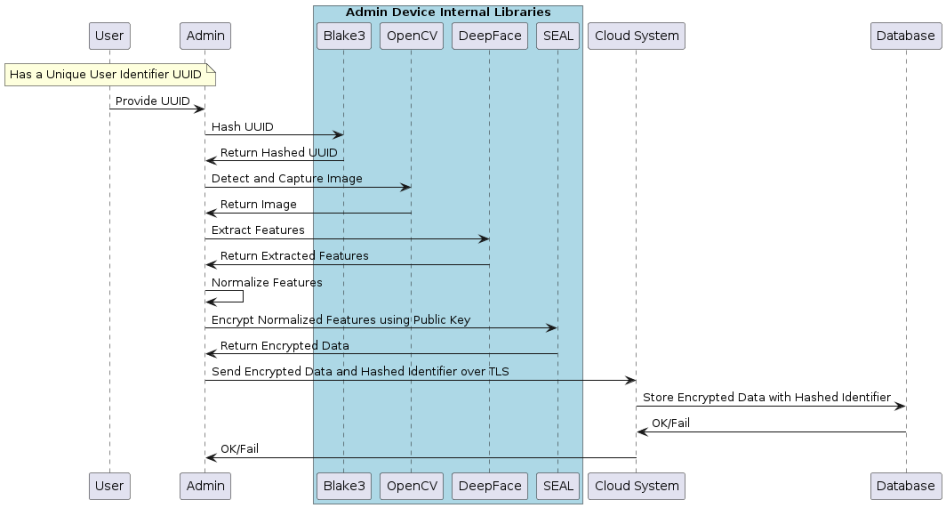


Figure 3.3: Enrollment process.

3.4.2 Revocation

The revocation process is a crucial security measure designed to remove access permissions, for instance, events of a security breach or simply the user no longer encompasses access rights. The sequence begins with the Admin, who holds a hashed version of the user's UUID.

When the revocation process is initiated, the Admin instructs the Cloud System to revoke the user's access permissions. The Cloud System then communicates

with the Database to remove all records associated with the user's UUID.

The Database executes the removal of the user data and returns a success or failure message to the Cloud System, which then forwards this outcome back to the Admin.

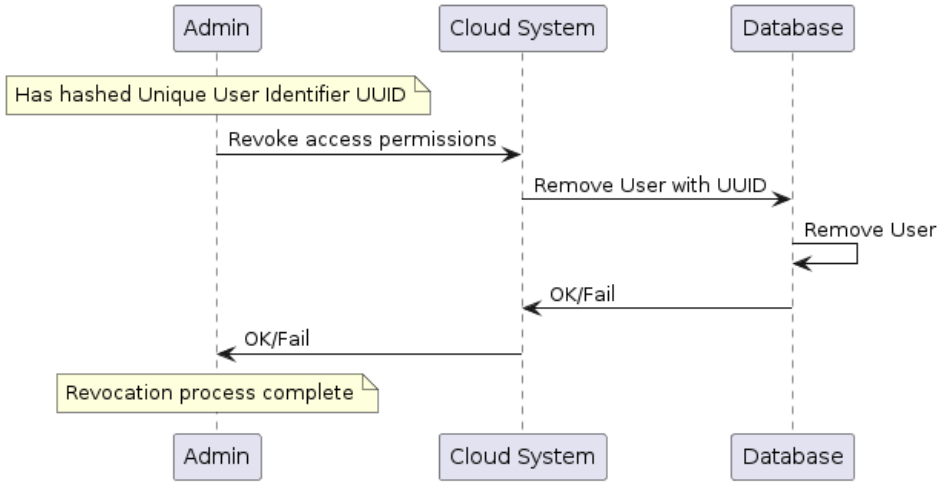


Figure 3.4: Revocation process.

3.4.3 Renewability

When there is a need for biometric updates, the renewal process is initiated by the Admin. The User then confirms the update and provides new biometric characteristics. The Admin verifies identities and coordinates with the Cloud System to check eligibility through the Database. If the user is eligible, meaning the user is already enrolled on the system. The Cloud System updates the biometric credentials and notifies the Admin of the outcome.

Credential renewal is a key requirement that the system needs to fulfil, which helps to protect against vulnerabilities and adapt to ongoing changes in biometric data, as people age and their biometric characteristics change with time.

3.5 IoT device

The IoT device is the primary interface for user identification claims and access control. The IoT device runs on a Linux operating system equipped with a camera and a subsystem that grants access if the identification process is successful.

3.5.1 Identification

The identification process is initiated when a user approaches the IoT device for an identity claim. The user provides its UUID and biometric characteristics to

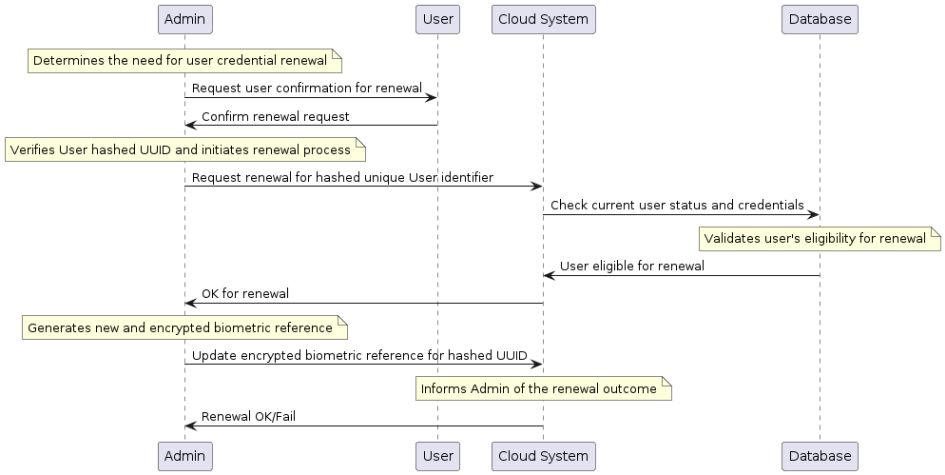


Figure 3.5: Renewal process.

the device. Subsequently, the device employs a model to extract features, normalises the data, and then encrypts it using the public key. The encrypted data is transmitted to the cloud over TLS communication.

3.6 Cloud system

The Cloud System's role is to store user credentials such as biometric references and perform homomorphic computations during identity authentication. Utilizing a cloud-based architecture allows scalability, where the biometric system can easily adapt to an increasing number of users and data without compromising performance. Moreover, the architecture ensures continuous availability whenever in need of access.

3.6.1 Operation under encryption

Performing operations under encryption allows us to calculate affinity and similarity metrics between encrypted vectors represented by the extracted and normalised biometric data. This approach is essential in preserving the confidentiality of user data throughout the computational process.

The CKKS scheme implemented on the SEAL library allows us to perform vector addition, subtraction and multiplication, corresponding to polynomial operations of encrypted numbers. It is possible to divide encrypted data by a constant value by multiplying it with its multiplicative inverse. However, SEAL inherently does not support division directly on encrypted data due to the lack of a method to translate the division of polynomials into corresponding operations under encryption. Some research papers try to solve this issue mathematically by approximating the division operation.

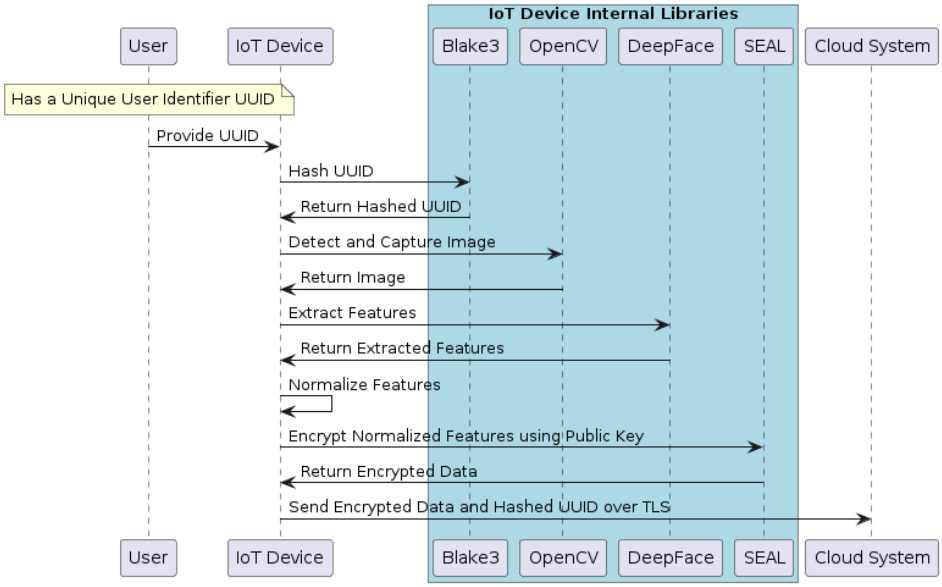


Figure 3.6: Identification process.

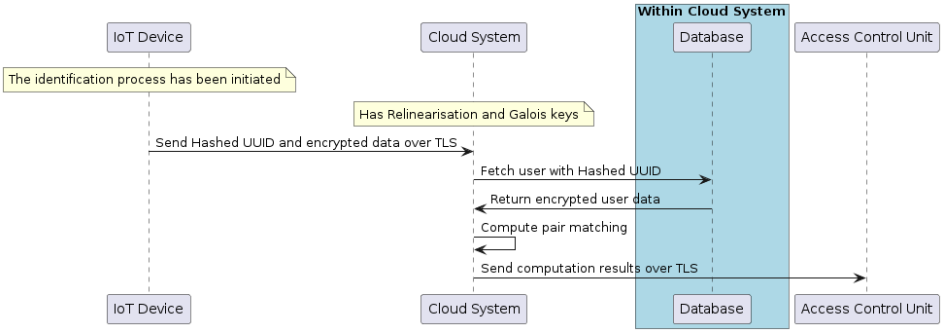


Figure 3.7: Cloud computation process.

Cosine Similarity

We employ cosine similarity to measure the orientation, meaning the angle between biometric references to assess their closely relatedness. To simplify cosine similarity computation under encryption, we have normalised the feature vectors representing user biometric reference before encryption. Namely, we only have to compute the dot product of the two encrypted vectors homomorphically and thus we don't have to perform division under encryption. The algorithm is presented below.

Algorithm 1 Dot Product

```
1: procedure DOTPRODUCT(x, y)          ▷ x and y are encrypted vectors.
2:     initialize: encProducts
3:     for i = 0 to size(x) − 1 do
4:         encProducts[i] ← x[i] ∗ y[i]
5:         relinearize(encProducts[i])
6:         rescale(encProducts[i])
7:     end for
8:     encSum ← encProducts[0]
9:     for i = 1 to size(encProducts) − 1 do
10:        rotated ← rotate(encProducts[i], i)
11:        encSum ← encSum + rotated
12:    end for
13:    return encSum
14: end procedure
```

Squared Euclidean Distance

Microsoft SEAL does not support the computation of square roots on encrypted data, which leads us to apply alternative approaches for operations that traditionally require this function. An alternative approach for calculating the Euclidean distance is to compute the Squared Euclidean Distance (SED) instead. This approach still achieves the primary objective of determining the magnitude or distance between two vectors in space and implies that we can compute the proximity between encrypted vectors. However, the ACU still needs to calculate the square root after decryption.

3.7 Access Control Unit

The Access Control Unit (ACU) handles the key generation described in 3.3, and the authentication process, which are the most pivotal parts of our privacy-preserving solution.

Algorithm 2 Squared Euclidean Distance

```
1: procedure SED(x, y)                    ▷ x and y are encrypted vectors.
2:     encDistSquared ← encrypt(0)
3:     for i ← 1 to n do
4:         encDiff ← x[i] − y[i]
5:         encSqDiff ← encDiff^2
6:         relinearize(encSqDiff)
7:         rescale(encSqDiff)
8:         encDistSquared ← encDistSquared + encSqDiff
9:     end for
10:    return encDistSquared
11: end procedure
```

3.7.1 Authentication

The authentication process is designed to ensure secure and controlled access based on encrypted data verification and threshold comparisons. This step plays a pivotal role in our access control solution, hence it controls what, when and where users should have access. The sequence begins with the Cloud System transmitting encrypted computed data to the ACU.

The ACU then checks the integrity and confidentiality shared by the Cloud System and proceeds with decryption. The ACU, utilising the SEAL library communicates with the Secure Element over a secure D-BUS communication to fetch the private key, which is provided with a limited duration to enhance security.

The core of the authentication mechanism is the threshold comparison performed by the ACU. It measures whether the decrypted data meets the predefined criteria. Depending on this evaluation, the ACU sends an outcome to the connected IoT Device, OK if the outcome is below the threshold or a Fail if above.

Upon receiving the result from the ACU, the IoT validates the integrity of the message and delegates a command to the Lock/Unlock subsystem within the IoT device. The subsystem either unlocks for a limited time, if the result is OK or remains in its default locked state if Fail. Subsequently, the subsystem provides feedback and informs the user of the current status.

The IoT device logs all identity claims and their respective outcomes to maintain a detailed audit trail, which can be used for troubleshooting.

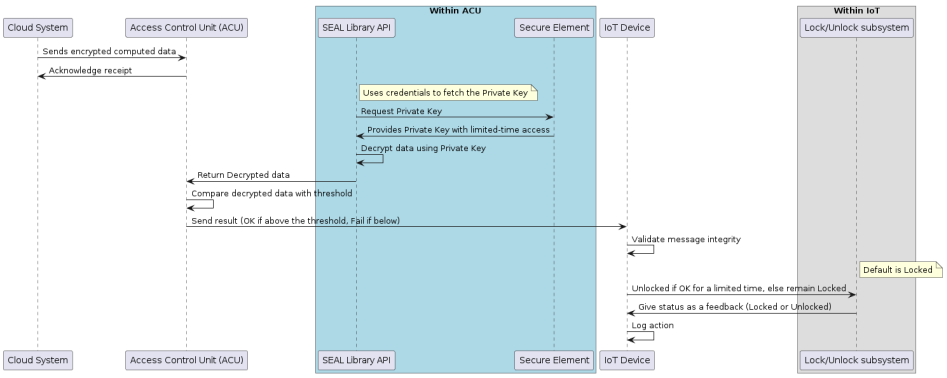


Figure 3.8: Authentication process.

Chapter 4
Results

This chapter aims to evaluate the performance of the biometric verification process by analysing the accuracy and reliability, operational efficiency and resource utilization. The evaluation will focus on key metrics such as the runtime for key generation, feature extraction and encryption, template retrieval and overall system runtime. Memory usage will also be analysed, given its significance in determining the scalability and deployability of biometric systems in different environments.

4.1 Biometric evaluation

4.1.1 Feature extraction

Feature extraction is done at the initial stages of the authentication process. This phase involves analyzing and converting an image containing facial features into a biometric template represented as a vector $\mathbf{z}$. Therefore, using 100 images from the LFW dataset we have presented the average run-time and standard deviation for the used recognition model in 4.1.

Model	μ_{IoT}	σ_{IoT}	μ_{Admin}	σ_{Admin}
GhostFaceNet	$8.34 \cdot 10^{-1}$	$1.39 \cdot 10^{-1}$	$1.07 \cdot 10^{-1}$	$2.18 \cdot 10^{-2}$
SFace	$1.56 \cdot 10^{-1}$	$3.56 \cdot 10^{-2}$	$1.91 \cdot 10^{-2}$	$2.70 \cdot 10^{-3}$
ArcFace	1.01	$1.95 \cdot 10^{-1}$	$8.57 \cdot 10^{-3}$	$1.62 \cdot 10^{-2}$
FaceNet	1.04	$1.67 \cdot 10^{-1}$	$1.22 \cdot 10^{-1}$	$2.46 \cdot 10^{-2}$

Table 4.1: Feature extraction for a recognition model in seconds.

4.1.2 Verification

The verification process is supervised and is based on a set of labelled image pairs. Each pair is marked with a binary identifier indicating whether the images are of the same person (True) or different (False). We then used the recognition models to extract features and computed the cosine similarity distance. We calculated the mean distance value and the standard deviation for the true positive and false positive predictions.

- True Positives $TP \sim \mathcal{N}(\mu_{TP}, \sigma_{TP}^2)$
- False Positives $FP \sim \mathcal{N}(\mu_{FP}, \sigma_{FP}^2)$

Where $\mathcal{N}$ represents the Normal or Gaussian distribution and μ, σ mean respective standard deviation.

The threshold x is the value at the intersection between the probability density functions of the TP and FP used to decide the authenticity of a user. It can be computed by solving the equation where the probability density functions are equal:

$$\frac{1}{\sigma_{TP}\sqrt{2\pi}} e^{-\frac{(x-\mu_{TP})^2}{2\sigma_{TP}^2}} = \frac{1}{\sigma_{FP}\sqrt{2\pi}} e^{-\frac{(x-\mu_{FP})^2}{2\sigma_{FP}^2}} \tag{4.1}$$

However, this is a difficult equation to solve numerically, thus we approach it empirically by optimising the threshold x that minimizes the absolute difference between the Gaussian distributions as:

$$x_{threshold} = \underset{x}{\operatorname{argmin}} \left| \frac{1}{\sigma_{TP}\sqrt{2\pi}} e^{-\frac{(x-\mu_{TP})^2}{2\sigma_{TP}^2}} - \frac{1}{\sigma_{FP}\sqrt{2\pi}} e^{-\frac{(x-\mu_{FP})^2}{2\sigma_{FP}^2}} \right| \tag{4.2}$$

After obtaining the threshold we determine the accuracy of the chosen models.

$$\text{Accuracy} = \left(\frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \right) \times 100\% \tag{4.3}$$

Cosine similarity

Model	μ_{TP}	σ_{TP}	μ_{FP}	σ_{FP}	$x_{threshold}$	Accuracy
GhostFaceNet	0.344	0.105	0.950	0.089	0.670	99.03%
SFace	0.297	0.093	0.862	0.101	0.569	99.35%
ArcFace	0.330	0.119	0.961	0.097	0.674	99.67%
FaceNet	0.19	0.058	0.909	0.139	0.41	99.67%

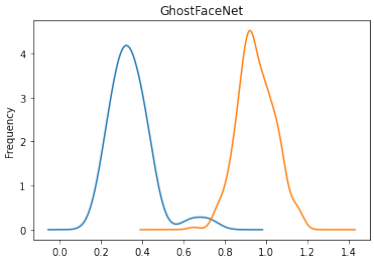
Table 4.2: Cosine similarity and observed accuracy of recognition models.

Euclidean distance

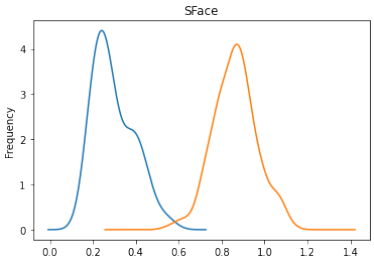
Model	μ_{TP}	σ_{TP}	μ_{FP}	σ_{FP}	$x_{threshold}$	Accuracy
GhostFaceNet	26.18	4.15	45.59	3.61	36.46	99.35%
SFace	8.109	1.127	13.76	1.25	10.81	98.06%
ArcFace	3.342	0.6654	6.0551	1.3903	4.455	99.03%
FaceNet	7.148	1.1383	15.7423	1.6176	10.772	99.35%

Table 4.3: Distance and observed accuracy of recognition models.

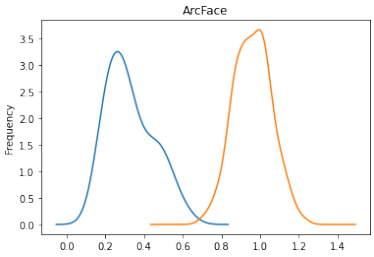
Humans can correctly distinguish faces with 97.53% accuracy [30].



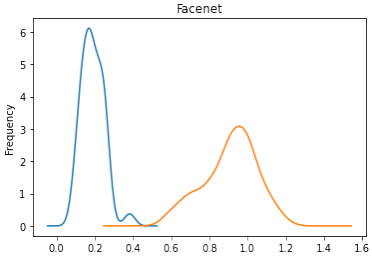
(a) GhostFaceNet.



(b) SFace.

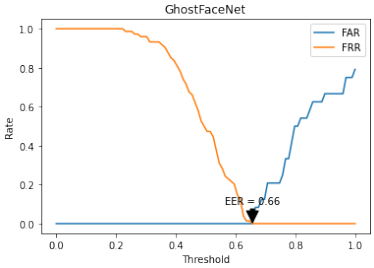


(c) ArcFace.

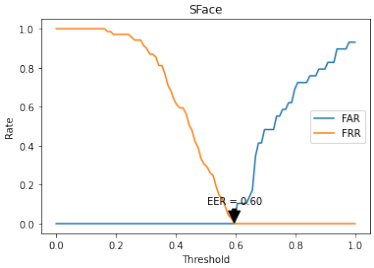


(d) FaceNet.

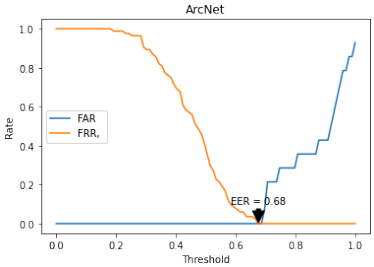
Figure 4.1: Cosine: True Positive and False Positive Distribution.



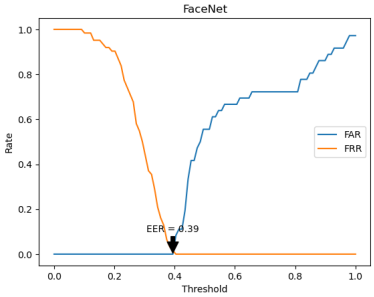
(a) GhostFaceNet.



(b) SFace.

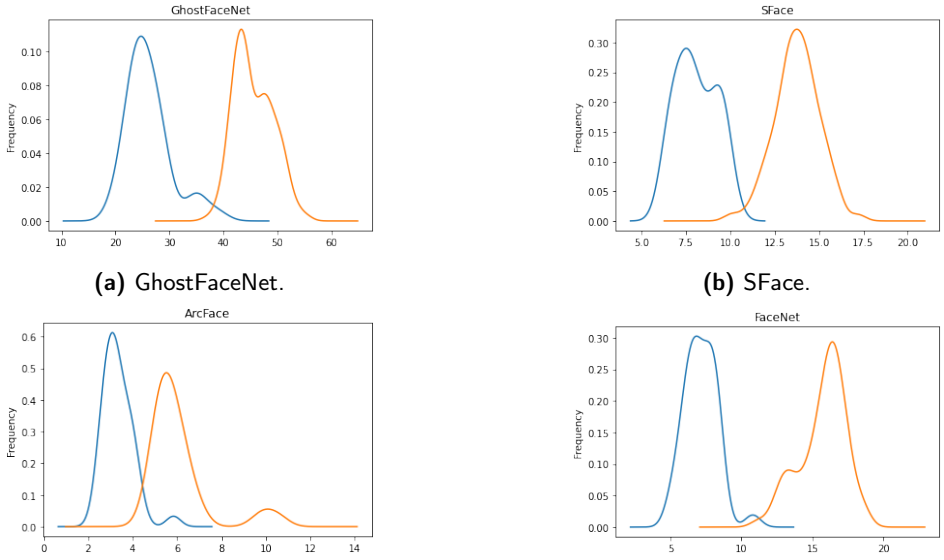


(c) ArcFace.



(d) FaceNet.

Figure 4.2: Cosine: FRR, FAR and EER.



(a) GhostFaceNet. **(b)** SFace.

(c) ArcFace. **(d)** FaceNet.

Figure 4.3: Euclidean: True Positive and False Positive Distribution.

4.2 Run-time and memory storage

The run-time benchmark was performed on Raspberry Pi 4 for key generation and encryption. The cloud homomorphic operations were done on a stationary desktop. The hardware specification of the machines used in this thesis looks as in 4.4.

Machine	Type	CPU	RAM
Raspberry Pi 4	IoT client and ACU	ARM Cortex-A72 64-bit 4-Core @ 1.5GHz	8GB LPDDR4-3200 SDRAM
Desktop	Admin and Cloud System	AMD Ryzen 9 5900X 64-bit 12-Core Processor @ 3.7 GHz	64GB DDR4 3200

Table 4.4: Hardware specifications of Raspberry Pi 4 and stationary Desktop.

4.2.1 Key generation

The ACU mainly handles the key generation process in our proposed solution. The parameters for the encryption scheme are experimentally chosen. 4.5

N	max bit-length	coeff-size	Δ
1024	27	{27}	-
2048	54	{27,27}	-
4096	109	{27,27,27,27}	2^{40}
8192	218	{60,40,40,60}	2^{50}
16384	438	{60,60,40,40,40,40,60,60}	2^{60}
32768	881	{60,60,60,60,40,40,40,40,40,40,40,40,60,60,60,60,60}	2^{60}

Table 4.5: Encryption parameters.

The results of the key generation for the secret, public, Galois and relinearisation are shown in the table below. 4.6

N	μ_{ACU}	σ_{ACU}	SK	PK	ReKey	GalKey
1024	-	-	-	-	-	-
2048	$2.92 \cdot 10^{-2}$	$6.80 \cdot 10^{-4}$	17KB	35KB	35KB	688KB
4096	$5.9 \cdot 10^{-2}$	$1.64 \cdot 10^{-3}$	69KB	137.6KB	412.6KB	9MB
8192	$5.46 \cdot 10^{-1}$	$6.9 \cdot 10^{-3}$	235KB	469KB	1.4MB	33.5MB
16384	4.971	$2.71 \cdot 10^{-1}$	918KB	1.8MB	12.8MB	336MB
32768	27.689	1.044	3.6MB	7.3MB	110MB	3.09GB

Table 4.6: Key generation in seconds with 100 iterations.

4.2.2 Encryption

Encryption run-time for the IoT device and Admin are illustrated in 4.7.

N	μ_{IoT}	σ_{IoT}	μ_{Admin}	σ_{Admin}	Ciphertext
1024	$1.82 \cdot 10^{-3}$	$3.42 \cdot 10^{-4}$	$3.48 \cdot 10^{-4}$	$4.08 \cdot 10^{-6}$	8.6KB
2048	$4.81 \cdot 10^{-3}$	$2.20 \cdot 10^{-3}$	$8.43 \cdot 10^{-4}$	$5.12 \cdot 10^{-6}$	17.3KB
4096	$1.36 \cdot 10^{-2}$	$1.73 \cdot 10^{-3}$	$2.36 \cdot 10^{-3}$	$1.89 \cdot 10^{-4}$	103KB
8192	$2.69 \cdot 10^{-2}$	$2.27 \cdot 10^{-3}$	$4.89 \cdot 10^{-3}$	$9.07 \cdot 10^{-5}$	334KB
16384	$1.07 \cdot 10^{-1}$	$6.19 \cdot 10^{-3}$	$1.68 \cdot 10^{-2}$	$9.92 \cdot 10^{-4}$	1.5MB
32768	$3.30 \cdot 10^{-1}$	$3.49 \cdot 10^{-2}$	$4.64 \cdot 10^{-2}$	$1.02 \cdot 10^{-3}$	6.8MB

Table 4.7: Encryption in seconds with 100 iterations.

4.2.3 Computation under encryption

This section shows the computational run-time for the dot product and Squared Euclidean Distance measurements under encryption.

Cosine similarity

We performed the dot product between the normalised and encrypted vectors to perform cosine similarity and the outcome is presented in 4.8.

N	μ_{Cloud}	σ_{Cloud}	Ciphertext
1024	-	-	-
2048	-	-	-
4096	2.525	$2.08 \cdot 10^{-2}$	69.3KB
8192	5.318	$6.06 \cdot 10^{-2}$	235KB
16384	20.815	$1.7 \cdot 10^{-1}$	1.3MB
32768	153.441	$2.89 \cdot 10^{-1}$	6.3MB

Table 4.8: Cosine operation in seconds with 100 iterations.

Squared Euclidean Distance

The results of the SED computation are shown in 4.9.

N	μ_{Cloud}	σ_{Cloud}	Ciphertext
1024	-	-	-
2048	-	-	-
4096	1.780	$1.43 \cdot 10^{-2}$	69.3KB
8192	3.512	$4.46 \cdot 10^{-2}$	235KB
16384	11.834	$1.03 \cdot 10^{-1}$	1.3 MB
32768	67.612	$4.64 \cdot 10^{-1}$	6.3MB

Table 4.9: SED operation in seconds with 100 iterations.

4.2.4 Decryption

The ACU executes the decryption process during user authentication and the run-time results are presented in 4.10.

N	μ_{ACU}	σ_{ACU}
1024	-	-
2048	-	-
4096	$4.07 \cdot 10^{-3}$	$5.57 \cdot 10^{-5}$
8192	$1.20 \cdot 10^{-2}$	$2.11 \cdot 10^{-4}$
16384	$4.38 \cdot 10^{-2}$	$5.36 \cdot 10^{-4}$
32768	$1.45 \cdot 10^{-1}$	$1.57 \cdot 10^{-3}$

Table 4.10: Decryption in seconds with 100 iterations.

Chapter 5
Discussion

This chapter will discuss the architectural design and implementation of a privacy-preserving biometric authentication system based on the Microsoft SEAL library and CKKS as an underlying homomorphic scheme. Moreover, we will reflect on utilising the ISO 2474:2022 standard as an outline for our client-to-server and server-to-client implementation and evaluate the biometric system.

5.1 Method

Implementing a client and server-based privacy-preserving biometric system as demonstrated in this thesis is feasible. The main challenge encountered during this thesis project is that it requires an extensive study to fine-tune which recognition model to choose. Also, selecting appropriate parameters for the CKKS scheme that give accurate results and acceptable run-time for user verification.

5.1.1 ISO:24745/2 objectives

Security

The CKKS scheme enables computation on encrypted biometric data without exposing the underlying biometric template. This ensures that the data remains encrypted throughout its lifecycle and thus protects against unauthorized access and maintains its confidentiality. Moreover, we rely on the TLS protocol for our client and server-based communication for data integrity.

The UUID proposed earlier works as a tool to enroll, identify, revoke, and renew user credentials. The UUID plays a pivotal role during the verification process, hence it is crucial when looking for a specific user already enrolled in the database. Furthermore, we rely on a cloud solution with reliable and distributed availability to mitigate risks such as DDoS attacks, power outages, and natural disasters. By leveraging cloud infrastructure, we ensure that the biometric information remains accessible across multiple geographic locations and operational environments.

Privacy

Feature extraction models offer a degree of irreversibility for biometric data. However, using the DeepFace library it's still feasible to deduce characteristics like gender, race, and age. Therefore, we rely on the CKKS scheme to provide a more secure form of irreversibility. This ensures that even if someone accesses the encrypted ciphertext, they cannot retrieve the original biometric template without the correct decryption key.

Furthermore, the scheme will result in two different ciphertexts when the same biometric input is provided. Hence, infeasible to link two encrypted templates belonging to the same individual and thus unlinkability is ensured.

The CKKS scheme also ensures that the biometric template remains encrypted. Only authorized entities with decryption keys can access the original data, securing personal information against unauthorized access and potential privacy breaches.

5.1.2 Dataset

We have chosen the LFW as our main dataset. Although the LFW dataset represents photos not taken in controlled environments there is still a lack of variability in lighting, poses, and expressions compared to recently created datasets such as Flickr-Faces-HQ **FFHQ**. Moreover, the LFW is getting outdated and contains images with low resolution.

Therefore, we evaluated the biometric system by combining images from the LFW dataset, photos from the DeepFace library and celebrity images freely available on the web in addition to the two datasets.

5.1.3 Facial recognition

We have chosen four facial recognition models with reported high accuracy to benchmark the biometric system. GhostFaceNet and ArcFace produce a vector $\mathbf{z}$ of size 1×512, while SFace and Facenet yield a vector of size 1×128.

We have obtained a high accuracy rate using the recognition models. However, this result should be taken with a grain of salt because we have used about 30 celebrities with labelled tests that verify the pairs. Altogether, we had 310 tests each test containing image pairs and a binary label indicating if the pairs are of the same person or not.

Furthermore, we actively chose not to implement VGG-Face because it outputs a vector of size 1×4096 and other facial recognition models were not selected because they had lower reported accuracy rates. GhostFaceNet is as of 2023 state-of-the-art recognition model tailored for IoT devices, which is reflected in table 4.1 when compared with other models.

5.1.4 Authentication

As mentioned earlier, it is feasible to implement a privacy-preserving biometric authentication system that can verify a user within five seconds, given the hardware setup described in the thesis. We have selected a polynomial degree of $N = 4096$,

and the scaling factor Δ is set to 2^{40}. Additionally, GhostFaceNet has been chosen as the recognition model. Below, we provide an estimate for the verification process.

$$\text{verification} = \text{feature extraction} + \text{encryption} + \text{computation} + \text{decryption} \quad (5.1)$$

$$\text{cosine} = 0.834 + 0.136 + 2.525 + 0.00407 = 3.49907 \approx 3.5 \text{ seconds} \quad (5.2)$$

$$\text{euclidean} = 0.834 + 0.136 + 1.780 + 0.00407 = 2.75407 \approx 2.76 \text{ seconds} \quad (5.3)$$

We have also concluded that $N = 1024$ and $N = 2048$ are not feasible to select as a polynomial degree due to their insufficiency for $\mathbf{z} = 1 \times 128$ and 1×512.

5.2 Threat analysis

The threat analysis discusses some threats and the proposed countermeasures and is presented on 5.1.

5.3 Ethics

Ethical considerations in this thesis is of high importance due to the sensitive nature of facial information. Although CKKS enhances privacy by encrypting data, the recognition models applied can still pose risks of privacy invasion if misused. Therefore, it's crucial to ensure that users explicitly consent to their participation in biometric systems and understand fully how their data will be used. Furthermore, organisations providing the authentication system must uphold transparent data handling practices to ensure user data privacy is respected.

Type	Threat	Countermeasure
T1	Access of private key	The private key is stored within a secure element and leased to the main process during the verification process.
T2	Fake biometric data during enrollement	Revoke and renew biometric reference for the compromised user.
T3	Spoofing facial detection procedure during identification.	Implement liveness detection algorithms to authenticate the presence of an actual person rather than an image representation.
T4	Compromised Admin	The ACU performs new key generation and all previous biometric templates are revoked.
T5	Compromise of stored data	Revoke and renew biometric template.
T6	Manipulation during comparison	Protect the comparison process using a secure TLS channel.
T7	DDoS	Analyse data traffic and limit the number of requests the system will accept from adversaries.
T8	Replay attack	Secure TLS channel.
T9	Man In The Middle Attack	Secure TLS channel.
T10	Biometric hill climb attack	Secure TLS channel.

Table 5.1: Threat analysis.

Chapter 6
Conclusion and Future Work

6.1 Conclusion

This thesis has successfully demonstrated the feasibility of a privacy-preserving biometric authentication system utilizing the CKKS homomorphic encryption scheme and the Microsoft SEAL library. We have structured a framework for both client-to-server and server-to-client communications by adhering to the ISO 2474:2022 standard for ensuring confidentiality, irreversibility, renewability, unlinkability and availability that is maintained throughout the biometric data's lifecycle. The thesis provided key findings in terms of the run-time benchmark for computation under encryption and also the accuracy metric of different recognition models.

6.2 Future work

Future work is to optimise the efficiency of the CKKS scheme, especially computation under encryption. More work is also needed to make approximations of non-linear functions like square roots for the CKKS scheme. Further research is needed to explore the integration of multimodal authentication methods that could enhance security and user convenience. This could include combining facial recognition with other biometric modalities such as fingerprint scanning or with multifactor authentication mechanisms like one-time passwords or traditional passwords.

References

[1] THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. 2016. *on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation.*. EUR-Lex. Available at: `https://eur-lex.europa.eu/eli/reg/2016/679/2016-05-04`

[2] Krzysztofek, M. 2021. *GDPR: Personal Data Protection in the European Union*. Wolters Kluwer Law International. ISBN: 9789403532707. EBOOK ISBN: 789403532714.

[3] Heinz, C. et al. 2021. *Privacy, GDPR, and Homomorphic Encryption*. Cham: Springer International Publishing. Available at DOI: `https://doi.org/10.1007/978-3-030-45316-9_8`

[4] Taylor J. 2019. *Major breach found in biometrics system used by banks, UK police and defence firms*. The Guardian Media. Available at: `https://www.theguardian.com/technology/2019/aug/14/major-breach-found-in-biometrics-system-used-by-banks-uk-police-and-defence-firms`

[5] ISO/IEC. 2022) *ISO/IEC 24745:2022 Information security, cybersecurity and privacy protection — Biometric information protection*. International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC)

[6] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. 1978. *ON DATA BANKS AND PRIVACY HOMOMORPHISMS*. Massachusetts Institute of Technology. Available at: `http://people.csail.mit.edu/rivest/pubs/RAD78.pdf`

[7] Upmanyu, M., Namboodiri, A.M., Srinathan, K., Jawahar, C.V. 2009. *Efficient Biometric Verification in Encrypted Domain*. In: Tistarelli, M., Nixon, M.S. (eds) Advances in Biometrics. ICB 2009. Lecture Notes in Computer Science, vol 5558. Springer, Berlin, Heidelberg. Available at DOI: `https://doi.org./10.1007/978-3-642-01793-3_91`

[8] Blanton, M., Gasti, P. 2011. *Secure and Efficient Protocols for Iris and Fingerprint Identification*. In: Atluri, V., Diaz, C. (eds) Computer Security – ESORICS 2011. ESORICS 2011. Lecture Notes in Computer Science, vol 6879.

Springer, Berlin, Heidelberg. Available at DOI: https://doi.org/10.1007/978-3-642-23822-2_11

[9] J. R. Troncoso-Pastoriza, D. González-Jiménez and F. Pérez-González. 2013. *Fully Private Noninteractive Face Verification.* in IEEE Transactions on Information Forensics and Security, vol. 8, no. 7, pp. 1101-1114, July 2013. Available at DOI: https://doi.org/10.1109/TIFS.2013.2262273.

[10] V. Naresh Boddeti. 2018. *Secure Face Matching Using Fully Homomorphic Encryption* 2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS), Redondo Beach, CA, USA, 2018, pp. 1-10. Available at DOI: https://doi.org/10.1109/BTAS.2018.8698601.

[11] G. Pradel and C. Mitchell. 2021. *Privacy-Preserving Biometric Matching using Homomorphic Encryption.* IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). Shenyang, China, 2021, pp. 494-505. Available at DOI: https://doi.org/10.1109/TrustCom53373.2021.00079.

[12] Kolberg J. 2021. *Security enhancement and privacy protection for biometric systems.* Doctoral dissertation, Hochschule Darmstadt, University of Applied Sciences. [online] pp.99-103. Available at: https://opus4.kobv.de/opus4-h-da/frontdoor/deliver/index/docId/234/file/PhDThesis_Kolberg.pdf

[13] Drozdowski P, Buchmann N, Rathgeb C, Margraf M, and Busch C. 2019. *On the Application of Homomorphic Encryption to Face Identification.* 2019 International Conference of the Biometrics Special Interest Group (BIOSIG), Darmstadt, Germany. Available at: https://christoph-busch.de/files/Drozdowski-FaceHE-BIOSIG-2019.pdf

[14] Paulin A. 2019. *Introduction to Abstract Algebra (Math 113).* University of California, Berkeley. Available at: https://math.berkeley.edu/~apaulin/AbstractAlgebra.pdf

[15] Bin Zhang. 2019. *A Remark on the Coefficients of Cyclotomic Polynomials.* Southeast Asian Bulletin of Mathematics. Vol. 43 Issue 4, p 615-618.

[16] Johansson T. 2006. *LECTURE NOTES IN CRYPTOGRAPHY 2006.* Lund University. Available at: https://www.eit.lth.se/fileadmin/eit/courses/edi051/lecture_notes/LN2.pdf

[17] Nigel P. Smart. 2016. *Defining Security.* Cryptography Made Simple. Available at: Doi https://doi-org.ludwig.lub.lu.se/10.1007/978-3-319-21936-3_11

[18] Craig Gentry. 2009. *A FULLY HOMOMORPHIC ENCRYPTION SCHEME.* Available at: https://crypto.stanford.edu/craig/craig-thesis.pdf

[19] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. *Homomorphic Encryption for Arithmetic of Approximate Numbers.* Advances in Cryptology. (ASIACRYPT 2017). Springer. pages 409-437. Available at DOI: https://www.doi.org/10.1007/978-3-319-70694-8_15

[20] Mohamad Alansari, Oussama Abdul Hay, Sajid Javed, Abdulhadi Shoufan, Yahya Zweiri, Naoufel Werghi. 2023. *GhostFaceNets: Lightweight Face Recognition Model From Cheap Operations*. Available at Doi: https://www.doi.org/10.1109/ACCESS.2023.3266068

[21] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, Stefanos Zafeiriou. 2018. *ArcFace: Additive Angular Margin Loss for Deep Face Recognition*. Available at Doi: https://www.doi.org/10.1109/tpami.2021.3087709

[22] Al-Taie, N. Azeez, A. Basbrain and A. Clark. 2017. *The Effect of Distance Similarity Measures on the Performance of Face, Ear and Palm Biometric Systems*.pp. 1-7. Available at Doi: https://www.doi.org/10.1109/DICTA.2017.8227495

[23] Fadi Boutros, Marco Huber, Patrick Siebke, Tim Rieber, Naser Damer. (2022). *SFace: Privacy-friendly and Accurate Face Recognition using Synthetic Data*. Available at Doi: https://doi.org/10.48550/arXiv.2206.10520

[24] Florian Schroff, Dmitry Kalenichenko, James Philbin. 2015. *FaceNet: A Unified Embedding for Face Recognition and Clustering*. Available at: Doi: https://doi.org/10.48550/arXiv.1503.03832

[25] 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.* University of Massachusetts, Amherst, Technical Report 07-49, October, 2007. https://vis-www.cs.umass.edu/lfw/

[26] Microsoft Research. 2023.*Microsoft SEAL*. Version 4.1. Available at: https://github.com/Microsoft/SEAL

[27] Jack O'Connor, Samuel Neves, Jean-Philippe Aumasson, Zooko. 2020. *Blake3*. Available at: https://github.com/BLAKE3-team/BLAKE3

[28] OpenCV. 2022. *Open Source Computer Vision Library*. Version 4.7.0. Available at: https://github.com/opencv/opencv

[29] Serengil, S.I. and Ozpinar A. 2020. *LightFace: A hybrid deep face recognition framework*. In: 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), [online] pp.23-27. Available at: https://doi.org/10.1109/ASYU50717.2020.9259802

[30] NADIA WHITEHEAD. 2014. *Face Recognition Algorithm Finally Beats Humans* . [online] Available at: https://www.science.org/content/article/face-recognition-algorithm-finally-beats-humans

Appendix A

CKKS encoding and decoding

A.1 CKKS

In this section, we will give an example of the full encoding and decoding procedure of the CKKS scheme.

A.1.1 Encoding

We want to encode the vector $\mathbf{z} = [1, 2]$ and the scaling factor Δ is 32.

Natural inverse projection

$$\pi^{-1} : \mathbf{z} \to \hat{\mathbf{z}} = [z_1, ..., z_{\frac{n}{2}}] \to [z_1, ..., z_{\frac{n}{2}}, \overline{z}_{\frac{n}{2}+1}, ..., \overline{z}_n] \tag{A.1}$$

and thus $\hat{\mathbf{z}} = [1, 2, 2, 1]$

Scaling

Here we multiply $\mathbf{z}$ with the scaling factor Δ.

$$\mathbf{z}' = \Delta \cdot \hat{\mathbf{z}} = [32, 64, 64, 32] \tag{A.2}$$

Lattice projection

The Vandermonde matrix A, found in 2.6 can be constructed using $\zeta = e^{\frac{2i\pi}{2n}} = e^{\frac{i\pi}{4}}$, hence the length n of our projected and scaled input vector $\mathbf{z}'$ is 4.

$$A = \begin{bmatrix} 1 & \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i & i & -\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i \\ 1 & -\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i & -i & \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i \\ 1 & -\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i & i & \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i \\ 1 & \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i & -i & -\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i \end{bmatrix} \tag{A.3}$$

Consequently, we obtain the coordinate vector $\mathbf{a}$ by using the formula given in 2.10.

$$\mathbf{a} = \left(\frac{192}{4}, -\frac{45.2548}{4}, \frac{0}{4}, \frac{45.2548}{4} \right) = (48, -11.3137, 0, 11.3137) \tag{A.4}$$

We round $\mathbf{a}$ to the nearest integer and get:

$$\mathbf{a} \approx [48, -11, 0, 11] \tag{A.5}$$

Thus, we get encoded integer polynomial $m(X) = 48 - 11X + 11X^3$

A.1.2 Decoding

Inverse scaling

We re-scale $m(X)$ by diving with Δ.

$$m(x)' = \frac{m(X)}{\Delta} = 1.5 - 0.3438X + 0.3438X^3 \tag{A.6}$$

Embedding

We evaluate the encoded polynomial $m(X)'$ on values that are roots of unity, i.e. $\zeta = (e^{\frac{i\pi}{4}})$.

$$\hat{\mathbf{z}} = [m(\zeta^1), m(\zeta^2), m(\zeta^3), m(\zeta^4)] = [1.0138, 1.9862, 1.9862, 1.0138] \tag{A.7}$$

Projection

In this step we reduce $\pi : \hat{\mathbf{z}} \rightarrow \mathbf{z}$

$$[1.0138, 1.9862, 1.9862, 1.0138] \rightarrow [1.0138, 1.9862] \tag{A.8}$$

Thus, we get our approximated input vector $\mathbf{z} = [1.0138, 1.9862]$.

Appendix B
Code examples

This chapter will demonstrate code examples for the feature extraction using DeepFace and then example applications of the CKKS scheme using Microsoft SEAL.

B.1 Feature extraction.

Listing B.1: Feature extraction example in Python.

```
from deepface import DeepFace
def extract_feature(path_to_image):
    return DeepFace.represent(path_to_image, "GhostFaceNet"
        )[0]['embedding']

z =  extract_feature("path_to_image")
```

B.2 Key generation

Listing B.2: Key generation in C++

```
#include <iostream>
#include <vector>
#include "seal/seal.h"
void generate_keys() {
    EncryptionParameters parms(scheme_type::ckks);

    size_t poly_modulus_degree = 4096;
    parms.set_poly_modulus_degree(poly_modulus_degree);
    parms.set_coeff_modulus(CoeffModulus::Create(
        poly_modulus_degree, {27,27,27,27}));
    auto context = SEALContext(parms);
    KeyGenerator keygen(context);
    PublicKey public_key;
    keygen.create_public_key(public_key);
    SecretKey secret_key = keygen.secret_key();
```

```
    RelinKeys relin_keys;
    keygen.create_relin_keys(relin_keys);
    GaloisKeys gal_keys;
    keygen.create_galois_keys(gal_keys);
}
```

B.3 Encryption

Listing B.3: Encryption in C++

```
#include <iostream>
#include <vector>
#include "seal/seal.h"
Ciphertext encrypt(SEALContext context, PublicKey
    public_key) {
    CKKSEncoder encoder(context);
    Plaintext plaintext;
    Ciphertext ciphertext;
    Encryptor encryptor(context, public_key);
    double scale = pow(2.0, 40);
    vector<double> z = {1, 2, 3, 4};
    encoder.encode(z, scale, plaintext);
    encryptor.encrypt(plaintext, ciphertext);
return ciphertext;
}
```

B.4 Dot product

Listing B.4: Dot product in C++

```
Ciphertext dotProduct(vector<Ciphertext> enc_v1, vector<
    Ciphertext> enc_v2) {
    /* CKKS parameters and necessary keys are generated*/

    /* Compute the element-wise products of the vectors */
    vector<Ciphertext> encrypted_products(enc_v1.size());
    for (size_t i = 0; i < enc_v1.size(); i++) {
        evaluator.multiply(enc_v1[i], enc_v2[i],
            encrypted_products[i]);
        evaluator.relinearize_inplace(encrypted_products[i
            ], relin_keys);
        evaluator.rescale_to_next_inplace(
            encrypted_products[i]);
    }
```

```
    /* Sum up the products using rotation */
    Ciphertext encrypted_sum = encrypted_products[0];
    for (int i = 1; i < encrypted_products.size(); i++) {
        Ciphertext rotated;
        evaluator.rotate_vector(encrypted_products[i], i,
            gal_keys, rotated);
        evaluator.add_inplace(encrypted_sum, rotated);
    }
return encrypted_sum;
}
```

B.5 Squared Euclidean Distance

Listing B.5: SED in C++

```
Ciphertext squaredEuclideanDistance(vector<Ciphertext>
    enc_v1, vector<Ciphertext> enc_v2) {
    /* CKKS parameters and necessary keys are already
        generated */

    /* Compute squared differences */
    vector<Ciphertext> squared_diffs;
    for (size_t i = 0; i < enc_v1.size(); i++) {
        Ciphertext diff;
        evaluator.sub(enc_v1[i], enc_v2[i], diff);
        evaluator.square_inplace(diff);
        evaluator.relinearize_inplace(diff, relin_keys);
        evaluator.rescale_to_next_inplace(diff);
        squared_diffs.push_back(diff);
    }

    /* Sum the squared differences */
    Ciphertext total_squared_dist = squared_diffs[0];
    for (size_t i = 1; i < squared_diffs.size(); i++) {
        Ciphertext temp;
        evaluator.add(total_squared_dist, squared_diffs[i],
            temp);
        total_squared_dist = temp;
    }

    return total_squared_dist;
}
```

B.6 Decryption

Listing B.6: Decryption

```
vector<double> decrypt(SEALContext context, SecretKey
    secret_key) {
    Decryptor decryptor(context, secret_key);
    Plaintext plaintext;
    decryptor.decrypt(ciphertext, plaintext);
    vector<double> result;
    encoder.decode(plaintext, result);
    return result;
}
```