

# Biometric data security using Homomorphic encryption

MSc Research Project  
Msc in Cybersecurity

Devansh Zaveri  
Student ID: x22194690

School of Computing  
National College of Ireland

Supervisor: Mr. Imran Khan

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**



**Student Name:** Devansh Zaveri.....  
**Student ID:** X22194690.....  
**Programme:** Msc in Cybersecurity..... **Year:** 2023 -2024  
**Module:** Msc Research Project.....  
**Supervisor:** Mr. Imran Khan.....  
**Submission Due Date:** 12/08/2024.....  
**Project Title:** **Biometric data security using Homomorphic encryption**  
**Word Count:** **Page Count :**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** .....Devansh Zaveri.....

**Date:** .....12/08/2024.....

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

## TABLE OF CONTENTS

Introduction.....	3
Research Questions.....	3
Literature Review.....	4
Research Method.....	6
A. Research Method.....	6
B. Planned Steps.....	7-8
Research tools and libraries.....	8
Design Specification.....	9
Implementation.....	10
Evaluation.....	13
Discussions.....	16
Conclusion and Future work.....	18
Reference List.....	19

# Biometric data security using Homomorphic encryption

Devansh Zaveri  
x22194690

## Abstract

In the modern tech world, biometric authentication technologies have become more and more common in a number of sectors, including mobile devices, banking, healthcare, and border control. However, there are serious security and privacy risks associated with the increasing reliance on biometric data. This is because biometric data is computed in raw formats on third-party cloud servers, creating privacy hazards as well as possibilities of illegal access to sensitive personal information. This paper explores the use of homomorphic encryption (HE) to safeguard user privacy and facilitate cloud-based computations on biometric data. Since cloud-based biometric authentication is becoming popular, it is necessary to handle this sensitive data securely. This study bridges the gap by designing a tailored HE algorithm that is perfect for biometric computations on cloud platforms. To find out how well our strategy maintains accuracy and privacy, we will compare it with other existing approaches. The primary feature of this encryption is that any kind of computation can be performed on encrypted data without changing its original format. The expected result is an observably secure and effective HE-based system for biometric calculations in the cloud, which will promote a wider usage of cloud authentication that preserves privacy.

**Keywords:** Secure Authentication and Processing, Cloud Privacy, Biometric Data, Homomorphic Encryption, Privacy-Preserving Computation

## 1 Introduction

The measuring of physiological and behavioral traits in humans, including voice, signature, and keystroke, as well as biological traits like fingerprints, faces, and irises, is known as Biometrics. The goal is to identify and characterize individual users. Because biometric features including resilience, invariance, and distinctiveness are desirable, biometric systems are widely used for identity verification in a range of applications, such as e-banking, e-health, and border control. Biometric recognition eliminates the downsides of traditional password- or token-based authentication techniques, such as the possibility of passwords being forgotten or guessed, and the possibility of tokens being lost or stolen. But, because of data privacy legislation like the CCPA and GDPR, it is now problematic to share private data like biometrics with unaffiliated groups, including cloud services or other enterprises. Businesses who violate these restrictions risk facing severe fines and damage to their reputation. Using conventional encryption techniques, sensitive data can be effectively and safely stored in encrypted form on cloud platforms. Nevertheless, businesses need to do computations on encrypted data by either decrypting the data in the cloud, which may lead to security risks, or download the data, decrypt it, perform calculations—a procedure that can be costly and time-consuming. By jumbling up the data, conventional encryption prevents it from being read without a secret key.

When data is decrypted, skilled hackers can also obtain the raw data, endangering privacy and integrity problems. Weak algorithms make it easy for expert hackers and sophisticated tools to decrypt data. Furthermore, processing on these encrypted data can completely alter the original format, even with robust encryption. Thus, these represent a few issues with the current systems. Therefore, the solution to this issue is to create an encryption algorithm that, even after calculating the cipher text, preserves the original data. Homomorphic encryption is a sophisticated technique that is presently being developed to do this. This algorithm is being updated daily and has not yet been fully applied in the actual world. Conventional encryption jumbles data, causing it unreadable. A unique feature added by HE is the ability to perform specific mathematical operations directly on the encrypted data. The term "homomorphic" refers to the preservation of mathematical properties. Consider the case where two numbers added before they are encrypted yield the same result as two numbers added after they are encrypted and decrypted. It's like doing calculations on encrypted files and never revealing the original information. Additional sections go over the entire encryption process in detail. By using HE, the cloud service provider or outsourcing business can only access encrypted data and process it before returning the encrypted result to the owner, who can use a private key to decrypt it. Sensitive data, such as financial transactions and medical records, can be analyzed by businesses and organizations without jeopardizing individual privacy. Our primary goal is to protect user privacy while offering businesses insightful information. Therefore, to sum up, our research aims to put a solution into practice by creating an algorithm that will encrypt biometric data and carry out various tasks like authentication, searching, pattern matching, aggregation, or data analysis while abiding by the CIA triad's principles.

**Question for Research.** The following research question is motivated by the above research problem:

In comparison to current techniques, is it possible to apply a homomorphic encryption scheme that will allow accurate biometric computation on cloud platforms while also clearly preserving user privacy and retaining the original form of data? And what kind of HE algorithm would work best against the calculations that would need to be made using biometric data?

## 2 Related Work

This section will examine existing cloud-based systems that utilize homomorphic encryption, highlighting their limitations. By comparing previous homomorphic encryption (HE) projects, we aim to identify shortcomings in current systems and propose innovative solutions or improvements in our research. We will subsequently present a detailed overview of relevant studies to inform our work. Also, it would demonstrate how our project uniquely stands out different as compared to these existing systems.

### ➤ A Review of Homomorphic Encryption for Privacy-Preserving Biometrics

Various studies, including those by Wang et al. (2023), Agrawal et al. (2019), and Naehrig et al. (2011), have explored the application of homomorphic encryption across sectors like healthcare, finance, and the Internet of Things (IoT). Homomorphic encryption enables the processing of biometric data while maintaining privacy, preventing the exposure of sensitive information when handling encrypted biometric templates (Wang et al., 2023; Li et al., 2018). However, for the practical implementation of homomorphic encryption in sensor networks, challenges such as computational overhead, key management, and scalability need to be addressed (Wang et al., 2023; Gentry, 2009). To tackle these challenges, ongoing research focuses on developing lightweight cryptographic primitives, implementing efficient key management protocols, and optimizing encryption algorithms (Wang et al., 2023; Naehrig et al., 2011).

### ➤ **Homomorphic Encryption Technology for Cloud Computing**

This paper discusses the theoretical underpinnings of homomorphic encryption, highlighting the pros and cons of partially, somewhat, and fully homomorphic encryption schemes. It also delves into the practical aspects of implementing homomorphic encryption in cloud computing environments. Key topics include performance overhead, computational complexity, and compatibility with existing cloud platforms. By comparing different homomorphic encryption schemes and their suitability for various applications, the authors illuminate the practical implications of deploying homomorphic encryption in real-world cloud computing scenarios (Min and Geng et al., 2019).

### ➤ **Optimal Multikey Homomorphic Encryption with Steganography Approach for Multimedia Security in Internet of Everything Environment**

Protecting the privacy and security of multimedia data transmitted over interconnected networks is critical in the rapidly evolving Internet of Things (IoT) and Internet of Everything (IoE) environments. Homomorphic encryption and steganography are two cryptographic techniques gaining significant attention for enhancing multimedia security in these settings. This literature review examines the current advancements and challenges in the field of optimal multikey homomorphic encryption combined with steganography for multimedia security in IoE environments. Steganography focuses on covert communication by embedding sensitive information within multimedia files. Numerous studies have explored the integration of steganography with homomorphic encryption to bolster multimedia security. The authors propose a steganographic method for embedding encryption keys within multimedia content to facilitate secure communication in IoE environments. However, issues related to the robustness and detectability of steganographic techniques remain, necessitating further research to address vulnerabilities and improve reliability (Abunadi et al., 2022).

### ➤ **Red Green Blue Image Encryption Based on Paillier Cryptographic System**

In this research, they offer a new method for encrypting red, green, and blue (RGB) images using the Paillier cryptography system. This method involves first dividing an RGB image into its individual channel images, and then applying the Paillier encryption function to the pixel intensity values. The encrypted image is then merged, and compressed if needed, and sent via an unprotected communication channel. After that, a decryption procedure recovers the sent image. The encrypted and recovered photos underwent a number of security and performance analyses to confirm their resilience to security breaches. However, because no calculations were made on the raw data, this encryption does not demonstrate homomorphism. The image encryption method only generates highly secured images. (Mamadou I Wade et al., 2018)

### ➤ **Role-based Access Using Partial Homomorphic Encryption for Securing Cloud Data**

A layer of secure cipher gateway for user data is introduced in this paper. In order to prevent unwanted access, this study presents an integrated system based on role-based access control principles and partial homomorphic encryption. To maintain data integrity, access to the data is strictly governed by the user-role mapping in the role-based hierarchy. Additionally, the suggested model offers mitigation strategies for different types of cloud threats. Based on a secure communication channel for data transfer with shorter encryption and decryption times, the work has been compared with its peers. By integrating RBAC with homomorphic encryption, the paper proposes a model that not only enforces strict access controls but also maintains data privacy even when accessed by authorized users. (Saxena, U.R., et al, 2023)

### ➤ **Paillier Cryptosystem Based Robust and Reversible Image Watermarking**

The paper "Paillier Cryptosystem Based Robust and Reversible Image Watermarking" explores an innovative watermarking method that ensures both robustness and reversibility, using the Paillier

cryptosystem. The Paillier cryptosystem is employed to encrypt the embedded image which ensures that the watermarking process is secure and the embedded image remains confidential. The proposed method is robust, meaning it can withstand attacks such as noise addition, and reversible, meaning the original image can be fully restored. It depicts a significant advancement in watermarking techniques by integrating robust and reversible methods. However, the use of the Paillier cryptosystem, while ensuring high security, might introduce computational overhead, potentially limiting its applicability in real-time or resource-constrained environments. Future research should focus on optimizing the computational efficiency of this approach.(Dash, Naik et al, 2024)

#### ➤ **Potential of Homomorphic Encryption for Cloud Computing Use Cases in Manufacturing**

The potential of homomorphic encryption for cloud computing in manufacturing sector is examined in this paper. Based on a review of the literature, the potential and restrictions for both homomorphic and classical encryption are first discussed. Second, simulations are run to verify the limitations by comparing the computation time and data transfer between homomorphic and classic encryption. The findings demonstrate that, depending on the use case, homomorphic encryption is a trade-off between security, time, and cost. Thirdly, use cases related to manufacturing are identified; the two use cases—predictive maintenance and contract manufacturing—are explained in detail and show the potential advantages of homomorphic encryption. (Kiesel, Marvin et al, 2023)

#### ➤ **A survey on implementations of various homomorphic encryption schemes**

Nevertheless, the current literature either ignores newly suggested HE schemes (like CKKS) or concentrates on a single kind of HE. Based on experimental results, we perform a thorough comparison and evaluation of the performance of homomorphic cryptosystems in this paper. All three HE families are covered in the study, along with a number of well-known schemes like BFV, BGV, FHEW, TFHE, CKKS, RSA, El-Gamal, and Paillier. The implementation specifications of these schemes in popular HE libraries, such as Microsoft SEAL, PALISADE, and HELib, are also covered. Furthermore, we examine the robustness of HE schemes against various forms of attacks, including integer factorization attacks on both classical and quantum computers and indistinguishability under specific plaintext attacks. (Doan,Tvt et al, 2023)

### **2.1 Summary table of the papers studied**

<b>Paper</b>	<b>Key findings</b>	<b>Methodologies</b>	<b>Contributions</b>
A Review of Homomorphic Encryption for Privacy-Preserving Biometrics	A number of HE methods related to biometrics were compared in terms of computational efficiency, to give readers a clear idea of each method's computing capacity. Potential methods for the study of HE were illustrated by a set of challenges and future research directions.	Systematic literature review and analysis of existing research. Evaluation of different HE schemes and their suitability for biometric applications. Benchmarking and performance evaluation of HE-based biometric systems. Analysis of emerging trends and research gaps in the field.	Provides a comprehensive overview of HE applications in biometrics, highlighting its advantages, challenges, and potential future directions. Provides a comprehensive overview of HE applications in biometrics, highlighting its advantages, challenges, and potential future directions.
Homomorphic Encryption Technology for Cloud Computing	HE enables secure cloud computing by allowing data to be	The paper evaluates different encryption schemes based on their	The paper offers a comprehensive overview of how HE can be applied to

	<p>processed in encrypted form. Fully Homomorphic Encryption (FHE) provides the strongest security guarantees but is often too computationally expensive for practical use in large-scale cloud environments. Partial or Somewhat HE schemes strike a balance between security and efficiency, making them more feasible for certain cloud applications like encrypted search or limited arithmetic operations. The paper highlights challenges in optimizing performance, ensuring scalability, and balancing encryption overhead in cloud settings.</p>	<p>ability to preserve data privacy in the cloud while allowing computations on encrypted data. The computational efficiency and performance overhead of HE in cloud environments are tested and compared across different encryption schemes. Real-world scenarios of HE applications in cloud services, such as secure data storage, encrypted searches, and encrypted machine learning, are explored.</p>	<p>enhance privacy and security in cloud computing. It provides guidance on optimizing HE to minimize computational overhead in cloud computing. The paper identifies key challenges for future research, such as improving the efficiency and scalability of FHE, as well as enhancing the usability of HE for cloud service providers.</p>
<p>Optimal Multikey Homomorphic Encryption with Steganography Approach for Multimedia Security in Internet of Everything Environment</p>	<p>The proposed Optimal Multikey Homomorphic Encryption with Steganography, model enhances multimedia data security in the Internet of Everything (IoE) environment. It provides superior results in terms of metrics like encryption quality and robustness compared to other approaches. The solution effectively addresses key challenges of secure data transmission, privacy, and processing limitations in IoE</p>	<p>Singular Value Decomposition (SVD) separates cover images into RGB components. Coyote Optimization Algorithm (COA) selects optimal pixels. Poor and Rich Optimization (PRO) with Multikey Homomorphic Encryption (MKHE) encrypts secret images. Steganography embeds encrypted data into cover images.</p>	<p>Combines homomorphic encryption and steganography for enhanced data security. Introduces optimization techniques (COA, PRO) to improve the accuracy of pixel selection and encryption quality. Provides a practical solution for multimedia security in resource-constrained IoE environments.</p>

	networks		
Red Green Blue Image Encryption Based on Paillier Cryptographic System	The Paillier cryptographic system effectively encrypts RGB images by applying homomorphic properties to each color channel (Red, Green, Blue) separately. High security for image encryption was achieved through probabilistic encryption, ensuring that the same pixel values are encrypted into different ciphertexts. The system introduces significant computational overhead, making it suitable for scenarios where security is prioritized over speed.	The Paillier encryption algorithm is applied to each of the three RGB channels independently. Each pixel value in the RGB image is treated as a plaintext and encrypted using the Paillier system. The Paillier system's homomorphic properties (addition over ciphertext) allow for operations on encrypted images without decryption. The encrypted images are evaluated for visual distortion, encryption time, and robustness against attacks.	First use of the Paillier cryptosystem for RGB image encryption, showing how homomorphic encryption can be adapted for multimedia data. Demonstrates how the probabilistic nature of Paillier encryption increases the security of image data by ensuring ciphertext diversity. Provides a benchmark for the computational costs and security benefits of using homomorphic encryption in multimedia encryption systems.
Role-based Access Using Partial Homomorphic Encryption for Securing Cloud Data	Role-based access control (RBAC) combined with Partial Homomorphic Encryption (PHE) enhances security for cloud data. PHE allows certain computations to be performed on encrypted data without needing decryption, making it efficient for implementing RBAC in cloud systems. The system is scalable to handle large data sets in cloud environments, allowing secure data sharing among users with different roles.	The paper integrates the RBAC model with Partial Homomorphic Encryption, ensuring users can only access the data they are authorized to view. A cloud-based system is designed and simulated to evaluate the efficiency and security of the proposed model. Performance metrics such as computational overhead encryption/decryption times, and access control enforcement are analyzed to validate the approach.	The paper proposes a novel framework combining RBAC with Partial Homomorphic Encryption for securing cloud data access. The work contributes to the development of an efficient and scalable access control system that can be practically implemented in cloud environments. By using PHE, the system allows for computations on encrypted data without compromising security, offering flexibility in handling different user roles.
Paillier Cryptosystem Based Robust and Reversible Image Watermarking	The proposed Paillier Cryptosystem-based watermarking is resilient to attacks such as compression,	The watermark is encrypted using the Paillier Cryptosystem, which supports additive homomorphism,	Introduced a novel combination of the Paillier cryptosystem with robust and reversible watermarking techniques,

	<p>cropping, and noise. The system allows for both the extraction of the watermark and perfect recovery of the original image, maintaining the integrity of the content. The system demonstrates moderate computational efficiency for real-time applications but can be optimized further.</p>	<p>allowing the encrypted watermark to be embedded into the image. A reversible data hiding technique is employed to ensure the original image can be fully recovered after the watermark is extracted. Various attacks like compression, noise, and geometric manipulations are applied to the watermarked images to test robustness. Metrics such as Peak Signal-to-Noise Ratio (PSNR) and Bit Error Rate (BER) are used to evaluate the watermarking's performance and image quality after extraction.</p>	<p>which is rare in image watermarking research. The system achieves both high security (via Paillier encryption) and reversibility (recovering the original image without loss). It establishes a foundation for future research on combining cryptographic methods with reversible watermarking, suggesting optimizations in terms of efficiency and scalability.</p>
<p>Potential of Homomorphic Encryption for Cloud Computing Use Cases in Manufacturing</p>	<p>HE enables manufacturers to perform computations on encrypted data in the cloud without revealing sensitive information, such as predictive maintenance or supply chain optimization. FHE provides strong security but has high computational costs, making it less practical and PHE may be more suitable for specific tasks. The main challenges include high computational overhead, key management complexities, and integration with existing manufacturing systems.</p>	<p>Identifies key use cases in manufacturing (e.g., predictive maintenance, quality control, supply chain management) where HE can enhance data security and privacy. Assesses the performance, security, and practicality of various homomorphic encryption schemes (FHE, PHE, SHE) for cloud-based applications in the manufacturing sector. Simulates a manufacturing process where encrypted data is used for secure cloud-based predictive maintenance, analyzing the trade-offs between encryption overhead and security.</p>	<p>Proposes a framework for using HE in cloud-based manufacturing environments to ensure secure data processing and storage. Identifies specific manufacturing use cases where HE can offer significant benefits, such as predictive maintenance, production optimization, and quality control, enhancing privacy and security in cloud-based environments. Recommends strategies for reducing computational overhead, optimizing key management, and improving the scalability of HE for broader use in the manufacturing sector's cloud computing ecosystem.</p>

A survey on implementations of various homomorphic encryption schemes	All schemes have varied performance and security guarantees. FHE offers the highest security but comes with computational overhead, whereas PHE and SHE provide a more practical balance for many applications, sacrificing some security for performance. While FHE is theoretically powerful, its practical use is limited due to performance issues. SHE and PHE schemes are more commonly implemented in applications like encrypted search and simple computations over encrypted data.	The paper surveys and classifies various homomorphic encryption schemes, across multiple implementation scenarios. The paper reviews specific implementations of HE schemes in real-world software libraries (e.g., HELib, SEAL, Paillier, etc.), evaluating factors such as encryption, decryption, computational complexity, and performance. Benchmarks are conducted to assess the efficiency, scalability, and applicability of each HE scheme across different types of computation and data sizes.	The paper provides a thorough survey of the current landscape of HE implementations, offering insights into the strengths and weaknesses of various schemes. By evaluating multiple HE schemes, the paper highlights practical recommendations for choosing the appropriate scheme based on the application needs (e.g., security, speed, or data size). It identifies gaps and challenges in the current HE implementations, encouraging further optimization in computational efficiency and applicability in resource-constrained environments.
---	--	---	--

### 3 Research Methodology

Users are able to work in untrusted environments and utilize encrypted data for computations without ever needing to be decrypted thanks to homomorphic encryption. Computing on encrypted data was first introduced by Rivest, Shamir, and Adleman in 1978. The application of homomorphic encryption (HE) to safeguard biometric calculations performed on cloud computing platforms is studied in this work. In this study, we would work with fingerprint biometrics and conduct some operations on them and deploy it on a cloud platform such as AWS. The growing popularity of cloud-based biometric authentication necessitates the development of strong methods for managing this private information while maintaining user confidentiality. The most important steps that would be taken during implementation are listed below.

#### 3.1 Research Method

##### 1. Collecting Fingerprint Data

We start by gathering fingerprint data from publicly available biometric datasets. Also we can collect the fingerprint data live from the fingerprint scanner which are stored as digital templates, following standards like ISO 19794-2 or ANSI 381. The files containing these digital fingerprints have the extension .EFT (Electronic Fingerprint Transmission). But we collected a few sample fingerprint images from publicly available github repository and used them for encryption.

## 2. Choosing the Homomorphic Encryption (HE) Scheme

Choosing the right type of encryption depends on the specific operations we want to perform on the fingerprint data. Different operations require different Homomorphic encryption algorithms. For example, simpler tasks like calculating distances, or just addition might only need Partial HE, while more complex tasks like feature comparison might require Somewhat HE or Fully HE. We have compared all the HE schemes below and choose a perfect one for our project. Before that, we would understand the basic concept of HE.

## 3. Basic Concept of Homomorphic Encryption and its types

Homomorphic Encryption works like a locked box that keeps the data inside safe while still allowing certain calculations to be performed on the outside. This means you can process the data without ever unlocking (decrypting) it.

Here's a basic example :

- Plaintext (source data): 10 plus 5
- Encryption: Within the homomorphic encryption box, you lock the digits 5 and 10. You encrypt these numbers and give it in the box.
- Computation on encrypted data: Without knowing the actual numbers, someone adds the data in the encrypted box.
- Decryption: After opening the encrypted box, the decrypted solution turns out to be: 15.
- What's amazing is that the computation was done without disclosing the initial values of 10 and 5. This is helpful in situations such as secure cloud storage, where you can store private data in the cloud and use it for computations without having to decrypt it.

### Types of HE schemes

#### ➤ Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption (FHE) is a powerful cryptographic method that allows computations to be performed directly on encrypted data. FHE uses advanced mathematical concepts like lattices and homomorphic polynomials for addition and multiplication. This ensures that even when data is processed on cloud platforms, it remains confidential. FHE supports the addition and multiplication necessary for fingerprint tasks like similarity matching, where encrypted fingerprint features can be compared, such as calculating the Euclidean distance. FHE supports arbitrary computations on encrypted data, meaning you can perform any number of additions and multiplications without ever needing to decrypt the data. This makes it extremely powerful but also computationally intensive. Examples include secure cloud computing, private machine learning, deep learning models and privacy-preserving blockchain applications. FHE can be used in environments like healthcare, finance, or government where maintaining the confidentiality of sensitive data during processing is crucial. Thus, FHE is the most powerful, enabling unlimited computations but is computationally expensive.

#### ➤ Partial Homomorphic Encryption (PHE)

PHE supports only one type of operation on encrypted data—either addition (e.g., Paillier encryption) or multiplication (e.g., RSA encryption), but not both together. It is useful in applications where only one type of operation is needed. For example, RSA can be used for secure signature verification, and Paillier for secure aggregation of data (like in electronic voting). PHE is often used as a building block in more complex cryptographic protocols, such as secure multiparty computation or zero-knowledge proofs. PHE

schemes typically focus on addition operations on encrypted data. In fingerprint recognition, features like minutiae points (ridge ends and bifurcations) are crucial for identification. PHE enables the comparison of encrypted minutiae templates while keeping the original data concealed. PHE is the most efficient but supports only one operation, making it suitable for simpler applications.

➤ **Somewhat Homomorphic Encryption (SHE):**

SHE stands for somewhat homomorphic encryption, which is the intermediate form between partially homomorphic encryption (PHE) and fully homomorphic encryption (FHE). SHE supports a limited number of operations on encrypted data, typically either a few multiplications or a combination of additions and multiplications. It is a compromise between efficiency and capability. SHE is suitable for situations where full FHE's computational overhead is too high, but some degree of homomorphic operations is needed, like in secure voting systems or encrypted search operations. SHE is less powerful, allowing limited operations, offering a balance between functionality and performance.

**In our project, our focus is to increase the brightness of an image which only involves adding a constant value to the pixel values of the image. Since Paillier homomorphic encryption (which is a type of PHE scheme) supports addition operations on encrypted data, we can use it to perform this task on encrypted file of our fingerprint image and deploy the project on AWS to demonstrate secure cloud processing. We do not require FHE and SHE as both are very inefficient for our requirement and would take a high computation time and would be expensive.**

#### 4. Deploying to cloud (AWS)

To deploy our Python project on AWS, we use Amazon EC2 for scalable computing power. Amazon Elastic Compute Cloud (EC2) is a web service provided by Amazon Web Services (AWS) that offers scalable and resizable computing capacity in the cloud. Essentially, it allows users to rent virtual servers also called as instances on which they can run applications, host websites, process data, and perform a wide range of other computing tasks without needing to invest in physical hardware. First, we created an EC2 instance and configure it with a suitable OS, such as Amazon Linux. We use SSH to securely connect to the instance and install necessary dependencies using pip, and set up any required environment variables. Finally, ensure your instance has the correct security group settings to allow inbound traffic on necessary ports, and start your application. Lastly, we clone the project files to the instance from github repository using git clone. This setup will allow our project to run on the cloud, leveraging AWS's robust infrastructure.

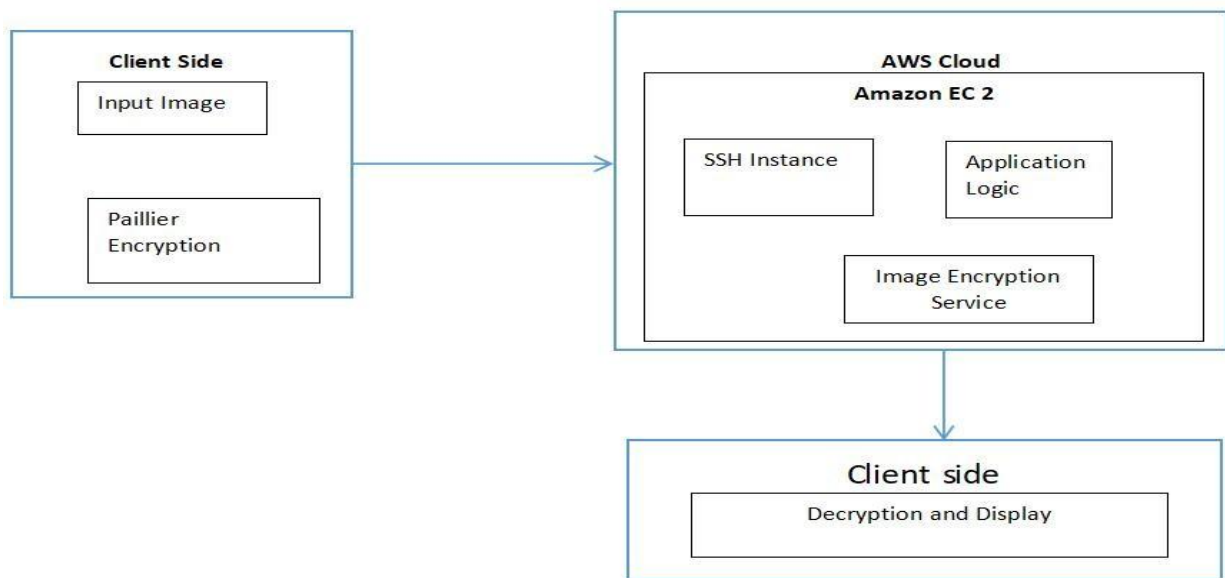
### 3.2 Research tools and Libraries used

An IDE and python programming language will be used to build our model which would encrypt the fingerprint files.

- **random** : The random module in Python provides functions to generate random numbers and perform random operations, such as shuffling a sequence. It includes methods for generating random integers, floating-point numbers, and choosing random elements from a list. In our algorithm, this random module is used to generate a random number 'r' which is coprime to 'n', where n is the product of prime numbers p and q. This 'r' is used to calculate the cipher text.
- **PIL (Pillow)** : Pillow is a Python Imaging Library that adds image processing capabilities to your Python interpreter. It allows for opening, manipulating, and saving many different image file formats. Commonly used for tasks like resizing, cropping, and adding filters to images. We use this library to open, show and save the images using the 'Image' module.

- **Matplotlib** : It is a powerful and widely used Python library for creating static, animated, and interactive visualizations. It offers a comprehensive set of tools for generating various types of plots, including, Line plots, Scatter plots, Bar charts, Histograms, Pie charts and many more. We have created the histogram to compare between the computed and the original image pixels.
- **NumPy** : NumPy (Numerical Python) is a powerful library in Python used for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is foundational for many other scientific and data analysis libraries in Python, such as SciPy, pandas, and TensorFlow. We first convert the image to array of pixels using NumPy before encryption and computation and again shape it back to graphical image format.

## 4 Design Specification



**Fig 1. Block Diagram of our Encryption Model**

### Description :

- At first, we have is the initial biometric image that we want to encrypt using Paillier homomorphic encryption. The client-side application (e.g., a Python script) is responsible for encrypting the image using the Paillier encryption algorithm.
- Next we have, the virtual server in the AWS cloud that will run the application responsible for handling encrypted images. The SSH Server allows secure remote connections to the EC2 instance. Then we have the Image Encryption Services running on the EC2 instance that handles image encryption, storage, and any homomorphic operations. In the application logic, we have the core Python application that performs the encryption, manages data, and handles requests.
- The client only sees the decrypted computed image on execution. Lets discuss the entire implementation in detail in next section.

## 5 Implementation

Implementing biometric image encryption using Paillier homomorphic encryption is a complex task that requires a deep understanding of cryptography, image processing, and cloud computing. Lets first consider the following key considerations and then understand the process step by step.

### Key Considerations

- **Biometric Data Sensitivity:** Handling biometric data requires strict security measures.
- **Paillier Efficiency:** The Paillier algorithm can be computationally expensive, especially for large images.
- **AWS Services Selection:** Choosing the right AWS services for storage, computation, and security is crucial.
- **Image Format and Preprocessing:** The image format and preprocessing steps will impact encryption and decryption performance. That is, if an image is of high resolution, it would take some time for encryption and decryption. Lets discuss each steps in detail :

### 5.1 Key Generation

Paillier is a probabilistic asymmetric encryption algorithm known for its homomorphic property. This means you can perform calculations on encrypted data without decrypting it first. This makes it useful in various applications like electronic voting, auctions, and secure data aggregation, biometric comparison, etc. In our project, we use paillier algorithm as it supports only addition which is our basic need and hence it is most suitable for adding constant factor to all pixel values and brightening the image.

#### How it Works:

- **Key Generation:** First it creates public and private keys using prime numbers.
- **Encryption:** It encrypts a message using the public key and a random number.
- **Decryption:** Decrypts the ciphertext using the private key.
- **Homomorphic Addition:** Adding two encrypted messages results in an encryption of the sum of the original messages.

#### Pseudo code to compute public and private keys :

```
function generate_keys(bitlength)
  p, q <- two large random primes of length bitlength/2
  n <- p * q
  lambda <- lcm(p-1, q-1)
  g <- random number in  $Z^{*n^2}$ 
  mu <-  $(L(g^\lambda \bmod n^2))^{-1} \bmod n$ 
  return public_key(n, g), private_key(lambda, mu)
```

#### Working :

##### ➤ **Generate Two Large Prime Numbers:**

Randomly select two distinct large prime numbers, denoted as p and q. The size of these primes directly impacts the security of the cryptosystem. Larger primes provide higher security but also increase Computational overhead.

##### ➤ **Calculate the Modulus:**

Compute the product of p and q to obtain the modulus n. This value will be part of both the public and private keys.

$$n = p * q \quad \dots\dots\dots(1)$$

### ➤ Calculate Euler's Totient Function:

Euler's Totient Function  $\phi(n)$  is defined as the number of integers less than  $n$  that are coprime (i.e., have no common divisors other than 1) to  $n$ . For a positive integer  $n$ ,  $\phi(n)$  counts the number of integers from 1 to  $n-1$  that do not share any prime factors with  $n$ . In other words, it counts how many numbers are coprime with  $n$ . If  $n$  is the product of two primes  $p$  and  $q$ , then  $\phi(n)=(p-1)\times(q-1)$ .

Determine the value of Euler's totient function for  $n$ , denoted as  $\phi(n)$ . This is calculated as:

$$\phi(n) = (p-1) * (q-1) \quad \dots\dots\dots(2)$$

To set up the Paillier cryptosystem, two large prime numbers  $p$  and  $q$  are chosen, and the modulus  $n$  is computed as  $n=p\times q$ . The value  $\lambda(n)$  is calculated using the least common multiple (LCM) of  $p-1$  and  $q-1$ . Since  $p$  and  $q$  are prime, Euler's Totient Function is used to compute  $\phi(n)$ , where:

$$\phi(n)=(p-1)\times(q-1) \text{ Then, } \lambda(n)=\text{lcm}(p-1,q-1) \quad \dots\dots\dots(3)$$

Select a Random Integer  $g$ :

Choose a random integer  $g$  where  $1 < g < n^2$ . This value must satisfy the condition that  $n$  divides the order of  $g$ . This can be efficiently checked using modular arithmetic properties.

### ➤ Calculate Lambda:

Compute  $\lambda = \text{lcm}(p-1, q-1)$ , where  $\text{lcm}$  stands for the least common multiple. \dots\dots\dots(4)

### ➤ Calculate Mu:

Calculate  $\mu = (L(g^\lambda \text{ mod } n^2))^{-1} \text{ mod } n$ , where  $L(x) = (x-1)/n$  and  $^{-1} \text{ mod } n$  represents the modular multiplicative inverse. \dots\dots\dots(5)

This complex calculation is essential for ensuring the correct decryption of messages. The value of  $\mu$  plays a crucial role in the decryption process, allowing the system to recover the original message from the encrypted ciphertext. The specific mathematical properties of this equation are carefully designed to ensure the security of the Paillier cryptosystem.

### ➤ Public and Private Keys:

The public key consists of the pair  $(n, g)$ .

The private key consists of the pair  $(\lambda, \mu)$ .

### ➤ Explanation of Parameters:

$n$ : The modulus, a composite number.

$g$ : A generator element.

$\lambda$ : Carmichael's totient function of  $n$ .

$\mu$ : A multiplicative inverse used in decryption.

### ➤ Security Considerations:

The security of the Paillier cryptosystem relies on the difficulty of factoring the modulus  $n$  into its prime factors  $p$  and  $q$ . The choice of large prime numbers is crucial to ensure the computational infeasibility of factoring  $n$ . Also, the proper random number generation is essential to prevent vulnerabilities in the key generation process. By following these steps and carefully selecting parameters, we can generate a secure Paillier key pair for encryption and decryption.

## 5.2 Encryption and Decryption

### ➤ Encryption

To encrypt a message  $m$  (where  $0 \leq m < n$ ) using the public key  $(n, g)$ :

Choose a random number: Select a random integer  $r$  such that  $1 \leq r < n$ .

Calculate the ciphertext: Compute the ciphertext  $c$  as follows:

$$c = (g^m * r^n) \bmod n^2 \quad \dots\dots\dots(6)$$

The ciphertext  $c$  is the encrypted form of the message  $m$ . Note that different values of  $r$  will result in different ciphertexts for the same message, adding randomness to the encryption process.

### ➤ Decryption

To decrypt a ciphertext  $c$  using the private key  $(\lambda, \mu)$  (from equations 2, 4, 5 and 6):

Calculate intermediate value: Compute  $d = c^\lambda \bmod n^2$ .

Apply  $L$  function on  $d$  where  $L$  is,  $= (n-1)/n$ . therefore we get,  $[c^\lambda \bmod n^2] - 1 / [c^\lambda \bmod n^2]$   
 $\dots\dots\dots(7)$

Recover the message: Compute the original message  $m$  as:

$$m = (L(d) * \mu) \bmod n \quad \dots\dots\dots(8)$$

The result  $m$  is the decrypted plaintext.

### ➤ Explanation of the Process

The encryption process involves raising  $g$  to the power of the message  $m$  and multiplying it with a random element  $r$  raised to the power of  $n$ . The result is taken modulo  $n^2$ .

The decryption process involves raising the ciphertext  $c$  to the power of  $\lambda$  modulo  $n^2$ , applying the  $L$  function, and then multiplying by  $\mu$  modulo  $n$  to recover the original message.

The security of the Paillier cryptosystem relies on the difficulty of computing  $m$  from  $c$  without knowing the private key  $(\lambda, \mu)$ .

## 5.3 Image processing before encryption and computation

This is the most crucial step of the implementation. We first convert the biometric image pixels to their corresponding numerical values (e.g., grayscale or RGB values). We then transform the image pixels which were in matrix form, into 'list' using numpy library of python. By creating this array, it becomes easy to encrypt list of pixels faster using a loop. We can also do image compression to reduce data size or resolution which makes all the computations faster. After encryption and computation, the array of pixels is reshaped back to graphical image.

## 5.4 Homomorphic encryption and operations

We encrypt each pixel integer using the Paillier public key and then perform necessary image computing operation that is addition to each pixel in their encrypted form. As, Paillier supports additive and multiplicative homomorphic encryption, we have performed additive HE as per our requirement which increases the pixel number and brightens the image.

## 5.5 Deployment on cloud - AWS

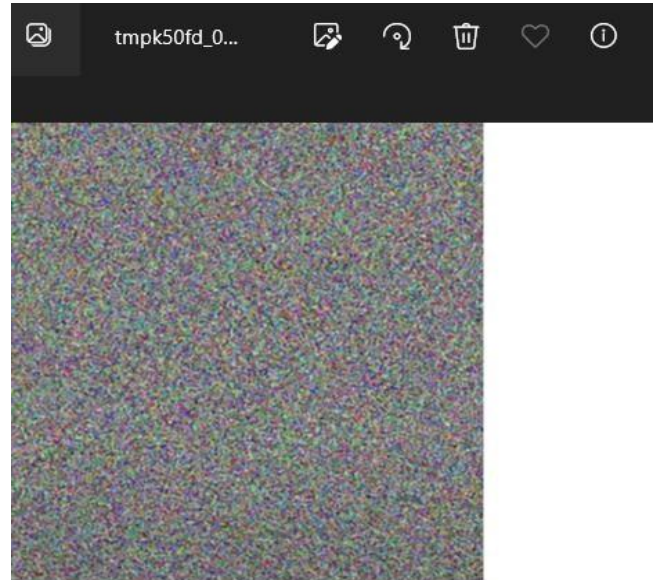
We create an AWS account and launch an EC2 instance. Now, choose an appropriate operating system i.e AWS Linux and configure security groups to allow SSH (port 22) and any other required ports for connection. A key pair for SSH access is downloaded in your local machine which we will need for authentication during secure login. Then update the package list and install Python and required libraries. After login in via ssh, we transfer the project files to the EC2 instance from github repository using git clone and execute it to perform encryption and computations. Thus this demonstrates secure data operations on cloud without hampering privacy.

## 6 Evaluation

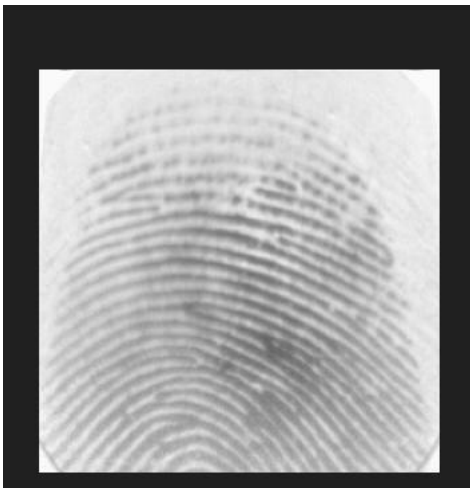
After seeing the implementation steps, let's dive into our project output and study in detail. We would then discuss and compare this with the other existing systems in the next section. In below figure 2, we see that the original thumb image is very dark and saturated. We first encrypt the image using paillier algorithm as explained above and we get the output as shown in figure 3. Now here comes the main part, where we perform the operation of brightness enhancement. First the pixels of the raw image are converted into the array of grayscale values using numpy library and its encrypted using paillier encrypt function. After encryption, we perform the homomorphic addition to each pixel, where values of each pixel is increased by factor 30 that brightens the image as shown in figure 4.



**Fig 2. Raw Fingerprint Image**



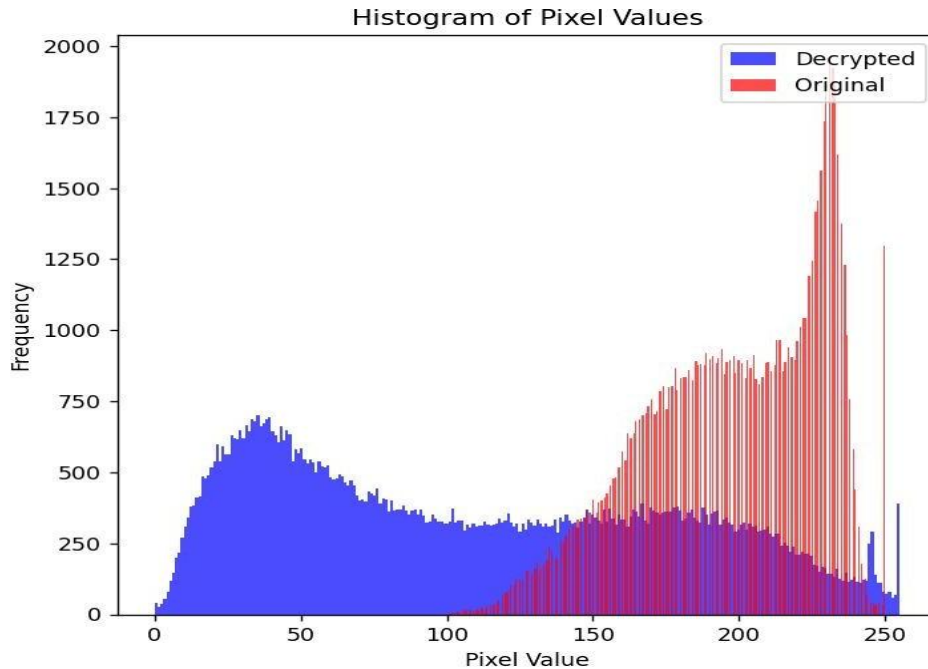
**Fig 3. Encrypted Image**



**Fig 4. Brightened Image**

Below in figure 5, is the histogram of the encrypted image that exhibits a shift towards higher pixel values compared to the original image, confirming the brightening effect. While pixel values are increased, it's crucial to assess whether the overall image structure and content are preserved after decryption and hence we find that our fingerprint image is same as that of original but in a brightened form. Any significant distortion or loss of information would diminish the effectiveness of the encryption method. The time taken to encrypt and decrypt images is also evaluated and it takes around 7-8 secs to get the final computed image.

But this measured time is with our image used in this project. The computation time can vary slightly depending on the resolution of the other images, because homomorphic encryption often comes with significant computational overhead. The Paillier cryptosystem's security properties is assessed in the context of image encryption and we find that it is very much suitable for image processing tasks. We do not require very complex algorithms like FHE for encryption, as our focus is to lessen the computation time as much as possible and provide the output.



**Fig 5. Histogram of pixel values**

We also calculated the mean and standard deviation that are statistical measures used to analyze the pixel values of images shown in fig 6. We do not require to calculate precision, recall or accuracy as we are not classifying or detecting anything here.

#### ➤ Mean

Mean (or average) is a measure of central tendency that provides the average value of a set of numbers. For pixel values in an image, the mean gives a sense of the overall brightness level. The mean pixel value is calculated by summing all pixel values and dividing by the total number of pixels. In our project, mean provides the average brightness of the original image and the decrypted image after processing. The mean pixel value of the original image versus the decrypted image can show how the brightness has changed due to the encryption and brightness adjustment.

#### ➤ Standard Deviation

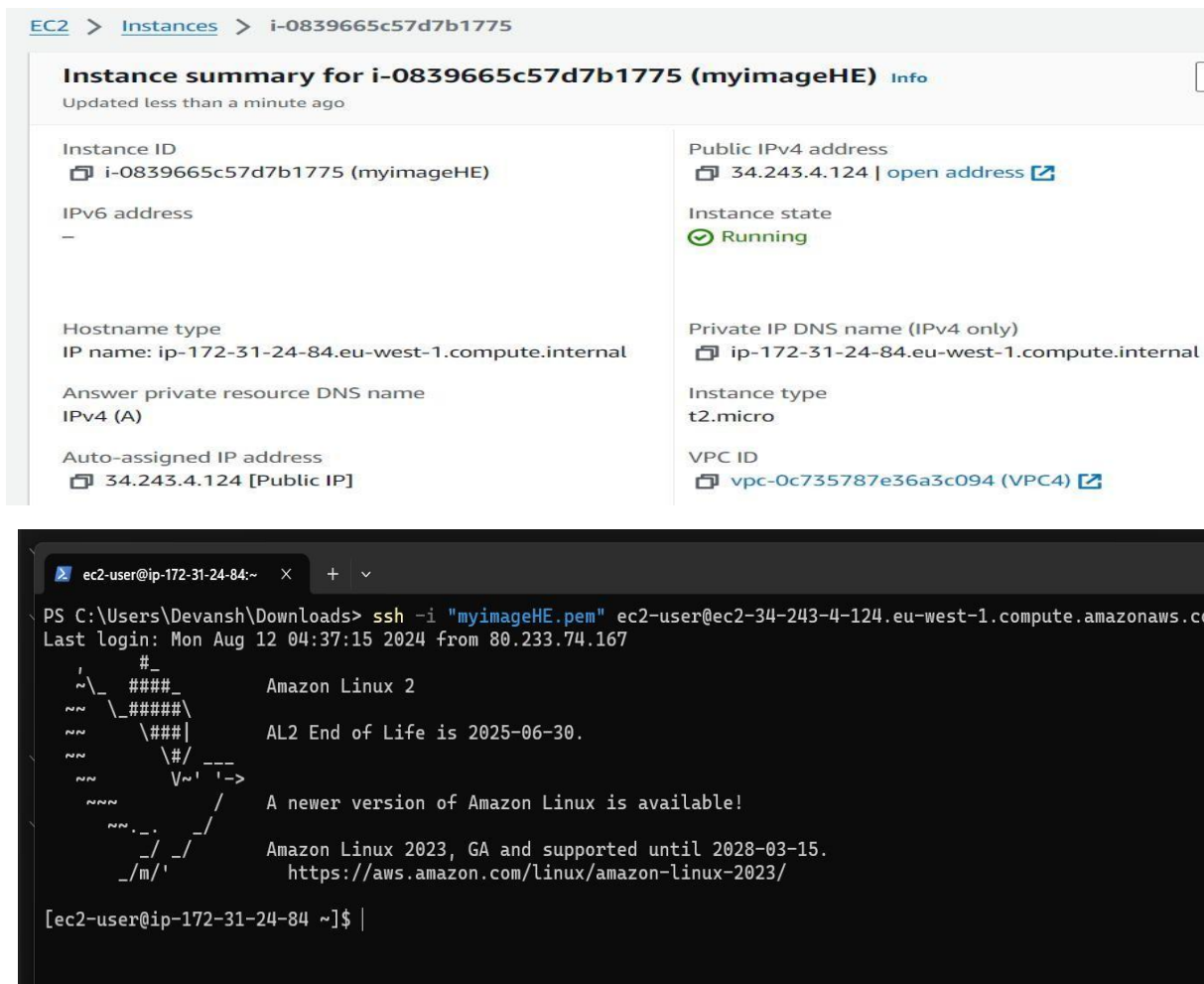
It is a measure of the dispersion or spread of a set of values. It indicates how much the pixel values deviate from the mean pixel value. The standard deviation is calculated as the square root of the variance. Variance is the average of the squared differences from the mean. The standard deviation of pixel values tells us about the spread of pixel intensities. A larger standard deviation indicates more variability or contrast in the image.

```
C:\Users\Devansh\AppData\Local\Programs\Python\Python37-32\python.exe C
Original Image - Mean: 108.59531111111112, Std Dev: 68.33012272305426
Decrypted Image - Mean: 198.30284444444445, Std Dev: 30.56541067427221

Process finished with exit code 0
```

**Fig 6. Mean and Standard Deviation of pixels**

We see that the mean value of the pixels of original image is 108.59 and that of Decrypted Image is 198.30. And the variance is 68.33 from the average value of pixels in original image and that for decrypted images it is 30.56. In next section, we would discuss and compare our project with the existing systems and the improvements needed.



The figure consists of two parts. The top part is a screenshot of the AWS Management Console showing the details of an EC2 instance named 'i-0839665c57d7b1775 (myimageHE)'. The instance is in a 'Running' state. Key details include: Instance ID: i-0839665c57d7b1775 (myimageHE), Public IPv4 address: 34.243.4.124, Instance state: Running, Hostname type: IP name: ip-172-31-24-84.eu-west-1.compute.internal, Private IP DNS name (IPv4 only): ip-172-31-24-84.eu-west-1.compute.internal, Instance type: t2.micro, VPC ID: vpc-0c735787e36a3c094 (VPC4).

The bottom part is a screenshot of a terminal window showing an SSH connection to the EC2 instance. The command used is 'ssh -i "myimageHE.pem" ec2-user@ec2-34-243-4-124.eu-west-1.compute.amazonaws.com'. The terminal output shows the Amazon Linux 2 logo and version information, including the end of life date (2025-06-30) and a newer version (Amazon Linux 2023) being available.

**Fig 7. Creating an AWS account and connecting to EC2 instance**

As shown in figure above, 'myimageHE.pem' helps us to authenticate into our ec2 and we are connected to our instance using ssh. We pull the project hosted on github using git clone and install all the python dependencies required. From below figure 8, we seen that we have successfully implemented our project on cloud ensuring secure image computation. Since AWS CLI, is a command line based, we cannot see the computed and encrypted image, and hence for this, we used a save function, so that it gets saved in the same directory with other project files after computation. Thus we hosted it on cloud and demonstrated secure processing without exposing the original biometrics.

```
[ec2-user@ip-172-31-24-84 ~]$ cd ImageHE/
[ec2-user@ip-172-31-24-84 ImageHE]$ python3 main.py
Private key generated is  $\lambda$  : 183742689236596893265893265893265893265893641678712615153648457575859959084

 $\mu$ : 21468732654782354872548725487254705060483216133079938281287109835507

Public Key :

n:      88041396365140909899757930477415234757203684478550234284295826001977761187771

Process finished with exit code 1
[ec2-user@ip-172-31-24-84 ImageHE]$ ls
101_4.jpg      encdec_func.py  histo.py      main.py      __pycache__  requirements.txt
computed.tif   encrypted.tif   keygenerate.py  primality_test.py  README.md    utility_functions.py
[ec2-user@ip-172-31-24-84 ImageHE]$
```

**Fig 8. Project running on AWS cloud**

## 6.1 Discussion

In this project, we explored the application of the Paillier cryptosystem for encrypting biometric images, specifically focusing on the process of image brightening. The project was deployed on AWS for cloud-based processing, leveraging the scalability and computational power of cloud infrastructure. This discussion will analyze the **findings, critique the experimental design, compare it to existing systems, and suggest potential improvements.**

### Major key Findings :

#### ➤ Image Brightening through Homomorphic Encryption:

The Paillier cryptosystem allowed for homomorphic addition, enabling the brightening of images by increasing pixel values while still in the encrypted domain. The encrypted image, when decrypted, showed a noticeable increase in brightness, demonstrating the feasibility of processing encrypted data without decrypting it. Deploying the project on AWS facilitated efficient processing of large biometric images. The scalability of AWS allowed for handling multiple image processing tasks concurrently, showcasing the practicality of cloud computing for intensive cryptographic operations.

#### ➤ Performance Metrics and Security :

The computational overhead of homomorphic encryption is significant, resulting in longer processing times compared to unencrypted image processing. Network latency and data transfer speeds impacted the overall performance when processing images in the cloud. The Paillier cryptosystem provided robust security for the biometric images, ensuring that the data remained confidential throughout the processing pipeline.

### Some critiques of our experimental design

#### ➤ Computational Overhead:

The primary drawback of the Paillier cryptosystem is its high computational cost. This was evident in the extended processing times for image operations. Existing systems that do not use encryption, or use less computationally intensive encryption methods, can perform similar tasks much faster. But where, security is a major concern than faster processing, some delay in processing is accepted.

#### ➤ Scalability and Efficiency:

While AWS provided the necessary infrastructure for scalability, the inherent inefficiency of homomorphic operations limited the overall performance. The project could benefit from optimization techniques such as parallel processing and hardware acceleration (e.g., using GPUs or specialized cryptographic processors).

The current design may not be suitable for real-time biometric applications due to the significant processing delays. Real-world systems require near-instantaneous processing, which is challenging with the current implementation of the Paillier cryptosystem.

This HE technique is not yet properly implemented in the real world and requires lots of modifications in their algorithms. Yet our project tries to demonstrate the use of basic HE to some extent, and to depict the securing of data on cloud platforms like AWS or any other third parties.

## Comparison to Existing Systems

### ➤ **Traditional Encryption Methods:**

Though Traditional methods like AES provide faster encryption and decryption and are very secure, but they lack the ability to perform computations on encrypted data. As when you compute on the encrypted data, it also changes the original format. The trade-off between security and efficiency is more balanced in traditional systems compared to the computationally heavy homomorphic encryption.

### ➤ **Other Homomorphic Encryption Schemes:**

Some fully homomorphic encryption (FHE) schemes, while even more computationally intensive, offer a broader range of operations than the Paillier cryptosystem. But this FHE scheme has the most complex mathematical calculations which would make it totally inefficient and would provide a huge latency, even for small computations. Thus this scheme was not suitable for our project. Partially homomorphic schemes, like ElGamal and Paillier, provide similar functionality but with different trade-offs in terms of security and efficiency.

**The existing systems have implemented HE in various other sectors as we saw in the literature review, but many of them have a huge latency for encryption, some of them have not perform any operations on encrypted data to test the homomorphism property. Its has become more necessary to deploy such algorithms to cloud companies first, rather than implementing in other sectors. Thus, our project demonstrates how securely we can carry out operations on encrypted format on a cloud platform, and how the original data remains secure and intact without being modified.**

## Suggested Improvements

- **Hardware Acceleration:** Utilizing GPUs or dedicated cryptographic accelerators could significantly speed up the processing of homomorphic operations. Implementing the project on FPGA-based solutions could provide a custom hardware approach to improve performance.
- **Hybrid Approaches:** Combining homomorphic encryption with traditional encryption methods could offer a balance between security and performance. For instance, using homomorphic encryption for critical operations and traditional encryption for less sensitive tasks.
- **Cloud Infrastructure Optimization:** Reducing network latency through edge computing or using AWS services optimized for low-latency data transfer could improve overall system performance. Implementing auto-scaling and load balancing features of AWS to dynamically allocate resources based on processing demands.

## 7 Conclusion and Future Work

In this project, we explored the application of homomorphic encryption to biometric image data using the Paillier algorithm, subsequently deploying the solution on AWS for secure and scalable processing. Our primary research question was to determine whether homomorphic encryption, specifically the Paillier algorithm, can be effectively utilized to secure biometric images while allowing meaningful computations to be performed on the encrypted data.

**Our most important objectives were threefold:**

- Implement the Paillier homomorphic encryption algorithm for biometric image encryption.
- Develop a system for performing necessary computations on encrypted biometric images without decrypting them.
- Deploy the developed system on AWS to ensure scalability and robustness.

Throughout this project, we successfully implemented the Paillier encryption algorithm to encrypt biometric images and created a framework for performing operations on this encrypted data. We then deployed this system on AWS, ensuring it could handle large volumes of data securely and efficiently.

**Key findings from our project include:**

- **Effectiveness of Homomorphic Encryption:** We demonstrated that the Paillier algorithm could be effectively applied to biometric images, allowing basic operations to be performed on encrypted data without compromising security.
- **Scalability and Performance:** By leveraging AWS, we ensured that our system could scale to handle large datasets and provide timely responses, crucial for real-world applications.
- **Security Assurance:** Our approach significantly enhances the security of biometric data by ensuring that it remains encrypted during processing, reducing the risk of data breaches.

The implications of our research are profound for fields requiring secure processing of sensitive biometric data. By ensuring that biometric images can be processed without decryption, we reduce the potential attack vectors and enhance the overall security posture of biometric systems. This advancement is particularly relevant in contexts such as secure access control, identity verification, and privacy-preserving biometric authentication systems.

But, our research is not without limitations. The primary limitations include:

- **Computational Overhead:** Homomorphic encryption introduces significant computational overhead compared to traditional encryption methods. This can impact the performance, especially for resource-intensive operations on large datasets. Also, for the very resolution images, like a 4k images, it take more computation time. The solution to this would be compressing the image first before processing on cloud.
- **Limited Operations:** The Paillier algorithm supports only a limited set of operations (addition and scalar multiplication) on encrypted data. This restricts the complexity of computations that can be performed homomorphically.
- **Scalability Constraints:** While AWS provides scalability, the cost associated with extensive use of cloud resources can be a limiting factor, especially for large-scale deployments.

### Future Works

- **Enhanced Algorithm Efficiency:** we can investigate more optimization techniques for the Paillier algorithm to reduce computational overhead. One of the primary limitations of homomorphic encryption is its computational intensity. Future research could focus on algorithmic improvements or hardware acceleration (e.g., using GPUs or specialized cryptographic processors) to enhance

performance. This would make the technology more practical for real-time applications, broadening its applicability and improving user experience.

- **Support for Complex Operations:** Integrate more advanced homomorphic encryption schemes like BGV or CKKS that support a wider range of operations, including multiplications. Paillier encryption is limited to addition and scalar multiplication. More complex operations are necessary for advanced biometric processing tasks. Expanding the range of operations would allow for more sophisticated analyses and applications, such as machine learning on encrypted data, further protecting user privacy while enabling rich data insights.
- **Hybrid Encryption Approaches:** Develop a hybrid approach combining homomorphic encryption with other cryptographic techniques like Multi-Party Computation (MPC) or Secure Enclaves. Thus by combining different cryptographic methods can leverage the strengths of each, offering enhanced security and performance. This approach could provide a more versatile and efficient solution, appealing to a wider range of applications and industries. The limitation of one algorithm can be covered by another.
- **Cloud-Based Encryption Services:** Develop a cloud-based service offering homomorphic encryption for biometric data as a service. Many organizations need secure biometric processing but lack the expertise or resources to implement it. Offering this as a managed service on platforms like AWS or other third party cloud companies could attract customers across sectors needing secure and compliant biometric processing. By this, we can commercialize a secure biometric authentication system for enterprises and consumers. There is a growing demand for secure authentication methods that protect user privacy, especially in financial services, healthcare, and government sectors. Therefore by providing a ready-to-deploy solution would meet the increasing need for secure, privacy-preserving authentication methods.

## References

Wade, Mamadou & Ogworonjo, Henry & Gul, Madiha & Ndoeye, Mandoye & Chouikha, Mohamed & Patterson, Wayne. (2018). "Red Green Blue Image Encryption Based on Paillier Cryptographic System."

Saxena, U.R., Alam, T. Role-based access using partial homomorphic encryption for securing cloud data. *Int J Syst Assur Eng Manag* **14**, 950–966 (2023). <https://doi.org/10.1007/s13198-023-01896-2>

Dash, A., Naik, K., Priyadarshini, P. (2024). Paillier Cryptosystem Based Robust and Reversible Image Watermarking. In: Namasudra, S., Trivedi, M.C., Crespo, R.G., Lorenz, P. (eds) Data Science and Network Engineering. ICDSNE 2023. Lecture Notes in Networks and Systems, vol 791. Springer, Singapore. [https://doi.org/10.1007/978-981-99-6755-1\\_34](https://doi.org/10.1007/978-981-99-6755-1_34)

Kiesel, Raphael, Marvin Lakatsch, Alexander Mann, Karl Lossie, Felix Sohnies, and Robert H. Schmitt. 2023. "Potential of Homomorphic Encryption for Cloud Computing Use Cases in Manufacturing" *Journal of Cybersecurity and Privacy* 3, no. 1: 44-60. <https://doi.org/10.3390/jcp3010004>

Doan, T.V.T., Messai, ML., Gavin, G. *et al.* A survey on implementations of homomorphic encryption schemes. *J Supercomput* **79**, 15098–15139 (2023). <https://doi.org/10.1007/s11227-023-05233-z>

Yang, Wencheng, Song Wang, Hui Cui, Zhaohui Tang, and Yan Li. 2023. "A Review of Homomorphic Encryption for Privacy-Preserving Biometrics" (2023) *Sensors* 23, no. 7: 3566. Available online at <https://doi.org/10.3390/s23073566>

E, Min & Geng, Yang. (2019). "Homomorphic Encryption Technology for Cloud Computing." *Procedia Computer Science*. 154. 73-83. 10.1016/j.procs.2019.06.012.

Abunadi, Ibrahim, Hanan Abdullah Mengash, Saud S. Alotaibi, Mashael M. Asiri, Manar Ahmed Hamza, Abu Sarwar Zamani, Abdelwahed Motwakel, and Ishfaq Yaseen. 2022. "Optimal Multikey Homomorphic Encryption with Steganography Approach for Multimedia Security in Internet of Everything Environment" *Applied Sciences* 12, no. 8: 4026. <https://doi.org/10.3390/app12084026>