

Configuration Manual

MSc Research Project
Cybersecurity

Devansh Zaveri
Student ID: X22194690

School of Computing
National College of Ireland

Supervisor: 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.....

Lecturer: 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:

Date:

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project, 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.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Devansh Zaveri
X22194690

1 Introduction

The configuration document outlines the procedures that must be followed in order to carry out the research project and provides the system setup needed in order to run the models. The crucial steps include setting up the required configuration for optimal operation, as well as obtaining and installing the necessary software and packages. Our project is entirely based on python and then deploying it to AWS cloud. Hence we require a high end laptop/computer, IDE, python libraries and AWS account. Other miscellaneous requirements are too listed below.

2 Experimental Setup

2.1 System Configuration

Hardware	Components	Specifications
HP Victus Laptop	Processor (CPU)	2.69 GHz 6-Core Intel Core i5
	Memory	16 GB RAM DDR4
	Storage	1TB Solid State Drive
	Operating System	Windows 11 Home Single Language - version (23H2)

2.2 Software (IDE)

PyCharm is a renowned Integrated Development Environment (IDE) specifically tailored for Python programming. Developed by JetBrains, it's a favorite among developers for its robust features, intelligent code completion, and seamless workflow. We are using the latest version of pycharm 24.1.6 we require following libraries to be installed first for our code to run.

- NumPy 2.0.1
- Matplotlib 3.9.1
- Scipy 1.14.0
- Phe 1.5.0
- Pillow 10.4.0
- Crypto 1.4.1
- Primality_test
- Utility_functions

The code is purely written in Python and is run in PyCharm IDE. We are using python interpreter 3.7 for the project.

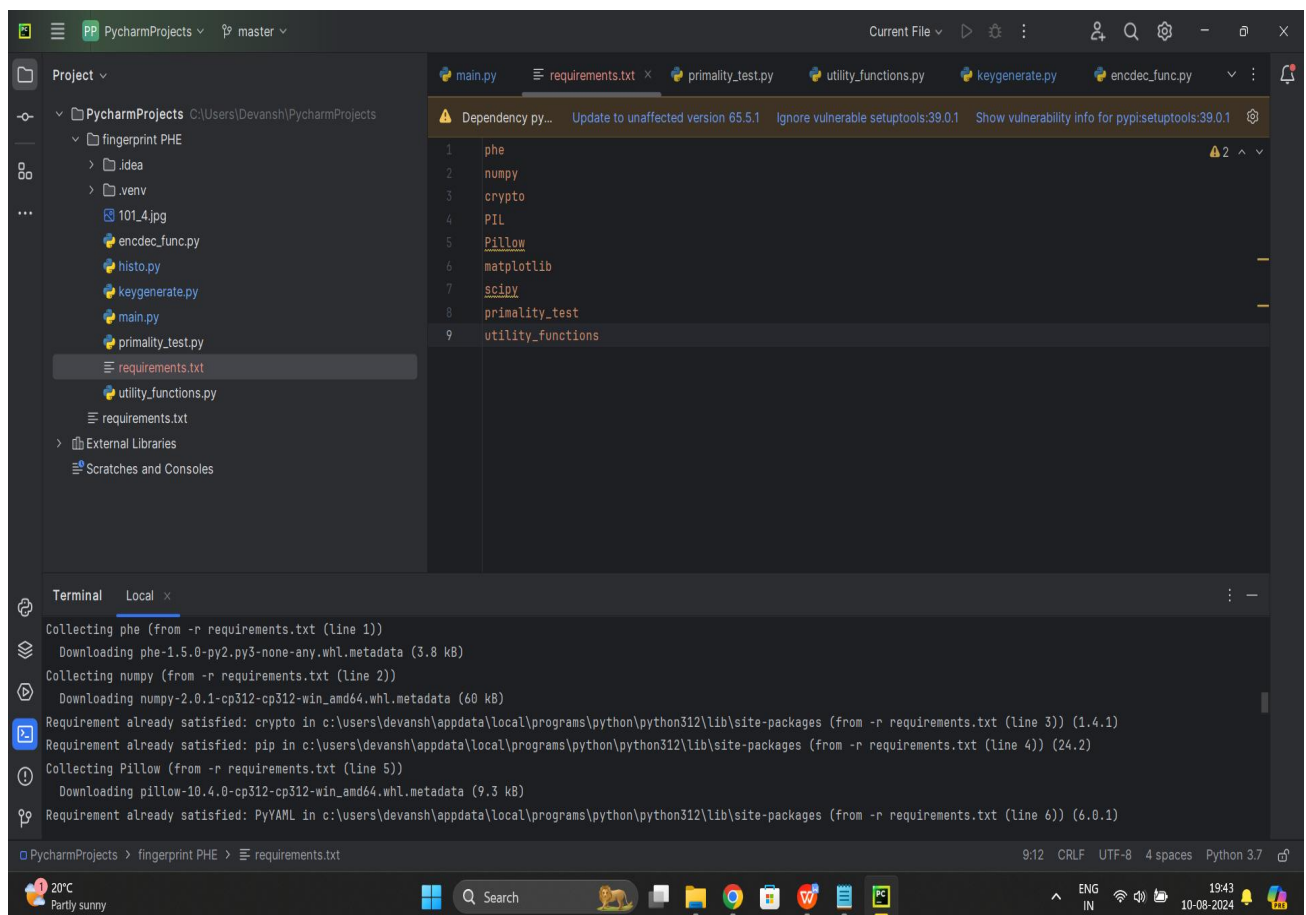
2.3 Dataset - (Fingerprint Images)

Fingerprint images are taken from the github as shown in the below source. GitHub has become a vast ecosystem for data sharing and collaboration along with the code repository. It has numerous datasets available now , helping data scientists, students and researchers to access a rich source of information for their projects. Though we only require one image at a time to process, we can process multiple images , one after other, in the project to see different processing times of different images.

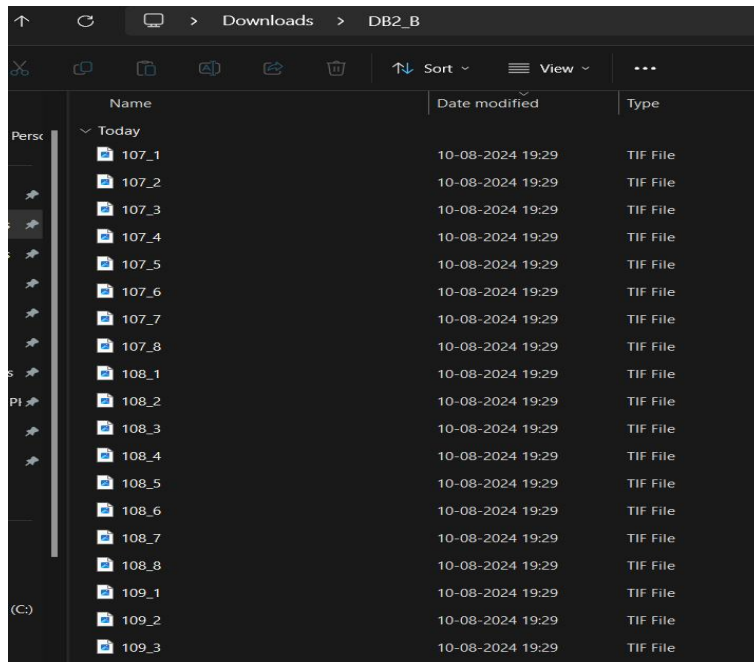
Source: <https://github.com/robertvazan/fingerprint-datasets?tab=readme-ov-file#fvc2004-db2-b>

3 Implementation Steps

3.1 Installing necessary python libraries using pip



3.2 Using fingerprint data



We have many fingerprints images to test and run on our code. We have selected one image out of these for testing. (101_4.tif)

3.3 Code Implementation

We have different functions built in different python files for calculating the public and private keys for encrypting and decrypting the image. After encrypting, operation of brightening the image is performed. The pixels of the image are first converted into array from the matrix format using numpy and each of them is converted to their corresponding pixel value. Each pixel value is then encrypted using public key. A new constant is added to each pixel value to increase its brightness and on decrypting, we get a new brightened image.



Original

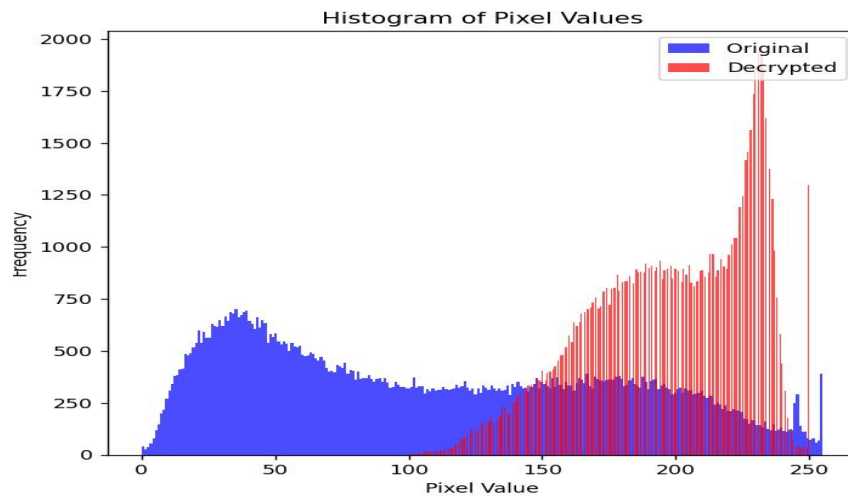


Encrypted



Brightened (Decrypted)

3.4 Image Comparison



Above is the histogram of the encrypted and decrypted image pixels. Brightened pixels value is more as shown in red as compared to the original dark image, shown in blue

4. Deployment on AWS cloud

AWS is a comprehensive cloud computing platform offering a wide range of services, including compute, storage, database, networking, analytics, and more. It provides on-demand access to scalable IT resources, enabling businesses to focus on core competencies without the overhead of managing infrastructure. Hence, the purpose of hosting our project on cloud, is to show the secure image processing on cloud platform without hampering any privacy. Here we use EC2 service by AWS for deploying our application and running it through AWS command line. The next important steps are listed below.

4.1 Launching the instance :

Amazon EC2 (Elastic Compute Cloud) is one of the core services provided by AWS. It offers scalable virtual servers in the cloud, known as EC2 instances. These instances can be used to run applications, host websites, process data, and more. EC2 provides a variety of instance types tailored to different use cases, such as general-purpose, compute-optimized, and memory-optimized instances. For our project we selected the following instance type.

General Purpose Free tier - t2.micro, Balanced CPU, 1 gb memory, and networking resources.

Next is choosing a machine image as described below. An Amazon Machine Image (AMI) is a template that contains the software configuration (OS, application server, and applications) required to launch an instance. When launching an EC2 instance, we choose an AMI that provides the operating system and pre-installed software needed for our project.

AMI chosen :

Amazon Linux 2: A lightweight Linux distribution optimized for cloud environments.

After configuring instance setting, a keypair file (.pem file) is downloaded in our local system. A Key Pair in AWS is a set of security credentials used for accessing EC2 instances. AWS uses public-key cryptography to secure the login process. When launching an instance, you are required to create or select a key pair. The private key is downloaded as a .pem file, which is used to SSH into your EC2 instance. Public Key is stored on the instance, used to encrypt login information and private key (.pem) is used to decrypt the login information and securely access the instance. If .pem file is lost, you cannot access your instance.

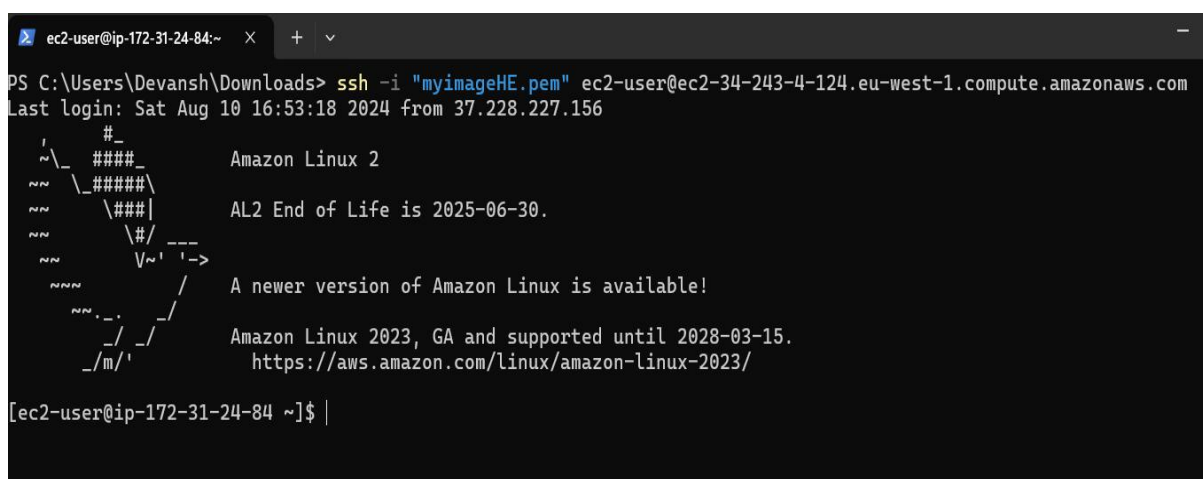
4.2 Configure security groups and Connecting to Instance

Security Groups act as virtual firewalls that control inbound and outbound traffic for your EC2 instances. You can define rules that allow or deny traffic based on IP addresses, protocols, and ports. Next is to set an Elastic IP which is a static public IPv4 address that you can allocate to your AWS account. It can be associated with any instance in your account, allowing you to maintain a consistent IP address even if you stop and start your instance.

Secure Shell (SSH) is a protocol used to securely connect to your EC2 instance.

Our ssh command to connect to our instance is :

`ssh -i "myimageHE.pem" ec2-user@ec2-34-243-4-124.eu-west-1.compute.amazonaws.com`



```
PS C:\Users\Devansh\Downloads> ssh -i "myimageHE.pem" ec2-user@ec2-34-243-4-124.eu-west-1.compute.amazonaws.com
Last login: Sat Aug 10 16:53:18 2024 from 37.228.227.156
#_
  _###_      Amazon Linux 2
 _###_
/###\
\###/
 \#/
  V~'  -->
  _/
  _/
 _/m/'
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-172-31-24-84 ~]$
```

Where , 34.243.4.124 is the elastic IPv4 address of our EC2 instance.

Final configurations of AWS EC2 for our project:

- **AMI:** Amazon Linux 2 AMI
- Instance Type:** t2.micro (Free Tier eligible) - Balanced CPU, 1 gb memory, and networking resources.
- **Key Pair:** myimageHE.pem
- **Security Group:**
 - Inbound: Allow SSH (22) from your IP, HTTP (80) from anywhere.
 - Outbound: Allow all traffic.
- **Elastic IP:** Allocated and associated with the instance. (Default setting)
- **User Data:** Install Python and necessary packages.
- **IAM Role:** Granted the instance necessary permissions to interact

5. Pulling our project to EC2 and running

After connecting to EC2 via SSH, pull our python project from our github repository using following command.

`git clone <our github project link>`. After the entire project is copied on EC2, install the necessary python libraries using `pip` command and run the project , same as we did in pyCharm.

```
ec2-user@ip-172-31-24-84:~/  + 
[ec2-user@ip-172-31-24-84 ImageHE]$ ls
101_4.jpg      histo.py      main.py      __pycache__  requirements.txt
encdec_func.py keygenerate.py primality_test.py README.md    utility_functions.py
[ec2-user@ip-172-31-24-84 ImageHE]$ python3 main.py
Private key generated is λ : 183742689236596893265893265893265893641678712615153648457575859959084

                                μ: 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]$ |
```

We run the `main.py` file and got the results. The computed and the encrypted images are saved in the file upon execution as AWS CLI do not support GUI, thus we cannot see the image. For this we use the `image.save()` function to which saves the images after they are processed.