d

National College of Ireland

Configuration Manual

MSc Research Project
Master of Science In Cyber Security

Anil Kumar Nagam
Student ID: X23196475

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Anil Kumar Nagam ……

Student ID: X23196475 ……

Programme: Master of Science In Cyber Security …… **Year:** 2024 ……..

Module: MSc Practicum …… ……..……

Lecturer: Jawad Salahuddin ……

Submission Due Date: 12/12/2024 ……

Project Title: Enhancing Data Privacy in Cloud Computing through Homomorphic Encryption ……

Word Count: 663 …… **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: Anil Kumar Nagam ……

Date: 12/12/2024 ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓
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.	✓

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

Anil Kumar Nagam
Student ID: X23196475

1 Introduction

The prototype demonstrates how homomorphic encryption can be used to protect sensitive data in cloud computing environments. It consists of three main modules:

- Encryption Module: Implements homomorphic encryption using the SEAL library.
- Cloud Integration Module: Allows secure data storage and retrieval from AWS, Google Cloud, and Microsoft Azure.
- Evaluation Module: Measures performance and security metrics of the encryption and cloud operations.

To run the prototype, you need the following:

1. Operating System: Windows 10, Linux (Ubuntu recommended).
2. Python Version: Python 3.8 or higher.
3. Libraries: The prototype requires several Python libraries, which are listed in the requirements.txt file.
4. Cloud Accounts: AWS, Google Cloud, or Azure accounts with appropriate access to cloud storage.
5. Hardware: A modern computer with at least 8 GB of RAM and 2 CPU cores.

2 Setup Instructions

Step 1: Install Python and Virtual Environment

Download and install Python 3.8 or higher from python.org.

Set up a virtual environment:

- Open your terminal (or command prompt) and navigate to the directory where you want to install the prototype.
- Run the following command to create a virtual environment:
  python -m venv homomorphic_env

Activate the virtual environment:

— On Windows:
homomorphic_env\Scripts\activate

— On Linux:
source homomorphic_env/bin/activate

Step 2: Install Required Libraries

Inside the activated virtual environment, run the following command to install the required libraries:

```
pip install -r requirements.txt
```

This will install all necessary dependencies, including SEAL (for homomorphic encryption), Boto3 (for AWS), Google Cloud Storage SDK, Azure Storage SDK, and others.

3 How to Use the Prototype

Step 1: Configuration

Cloud Setup: Configure your cloud account credentials for AWS, Google Cloud, or Azure:

- For AWS, create an IAM user with appropriate permissions and generate access keys.
- For Google Cloud, create a service account and download the credentials JSON file.
- For Azure, generate a connection string for your storage account.

Modify Configuration Files:
In the cloud_integration/cloud_setup.py file, input your cloud credentials:
- For AWS, set aws_access_key_id and aws_secret_access_key.
- For Google Cloud, set the path to your service account credentials JSON file.
- For Azure, set your connection string.

Step 2: Running the Encryption

The prototype uses homomorphic encryption to secure data. In the homomorphic_encryption.py file, you can specify the data to encrypt.

The encrypt() method encrypts input data, and the decrypt() method returns the decrypted data.

4 Running the System

Step 1: Running the Full System
To run the full encryption, upload, and decryption process, execute the main.py file:

```
python main.py
```

This will:
- Encrypt a sample data string.
- Upload the encrypted data to the selected cloud platform.
- Retrieve and decrypt the data.
- Print the performance and security evaluation results.

5 Evaluation

The prototype includes an evaluation module that measures:

1. Encryption and Decryption Time: This is the time taken to encrypt and decrypt data.
2. Upload and Download Time: Measures the cloud storage interaction times.
3. Security Assessment: Ensures that the system maintains data integrity and resists unauthorized access and tampering.

To run the evaluation, simply execute the performance_metrics.py and security_assessment.py scripts as part of the main.py execution.

6 Troubleshooting

Common Issues:

- Cloud Credentials Not Working:
  Double-check that your cloud credentials are correctly configured in the cloud_setup.py file.
  Ensure that you have appropriate access permissions for your cloud storage service.

- Encryption Errors:
  Ensure that you are using the correct format for the input data.
  If the encryption process fails, verify the configuration of the SEAL library.

- Slow Performance:
  The encryption and decryption processes are computationally intensive. Test with smaller datasets to ensure the system is working correctly.
  Consider optimizing the encryption parameters if running large datasets.

7 Conclusion

This prototype demonstrates how homomorphic encryption can be integrated with cloud computing to ensure data privacy without sacrificing usability or performance. By enabling secure data storage and processing across multiple cloud platforms, the system provides a powerful solution for industries that require high levels of data security. Further optimization and testing can enhance the system’s scalability and efficiency.