**I need an interim homomorphic demo built based on the specification below. I have chosen to use the SEAL homomorphic encryption library so it will probably have to be coded in C++. Below is how you could develop the interim demo to demonstrate uses and safety of homomorphic encryption.**

PROJECT ID: 40

Title: Homomorphic Encryption Demo

Description

Homomorphic encryption is a cutting-edge type of cryptography that enables computation on encrypted data without the need to decrypt. For data analysis, this can solve many privacy and security concerns. However, homomorphic encryption (HE) comes with additional challenges; homomorphic encryption schemes are typically computationally complex and not currently feasible for real world computation scenarios. Further efforts are placed on exploring novel implementation strategies to accelerate HE. This project investigates the possibility of HE in a real-world environment, showcasing the potential of and the challenges associated with homomorphic encryption. We have ongoing research in this field and a basic demo has been created within the ECIT Engineering team at CSIT, QUB. The objectives of this project are to become familiar with homomorphic encryption and be able to build an existing homomorphic encryption library (e.g. SEAL or OpenFHE) and demonstrate performance of HE, to propose a working demo of HE targeting a suitable use case, and to evaluate the capabilities and limitations of the HE demo. This project can focus either on the core functionality and use case choice, or can also explore the development of frontend interface to enhance the demo quality and functionality, with the option to feed into and build on the existing demo provided at CSIT. Students considering this project should have an interest in cryptography, research and some software programming experience.

Keywords: cryptography; encryption; demo; encryption; data security

Available for 2 students

**Step 1: Set Up the Development Environment**

• Install the SEAL Homomorphic encryption library. and tools for homomorphic encryption in your programming environment.

**Step 2: Implement Basic Homomorphic Operations**

• Start with the basics: key generation, encryption, and decryption. Create functions for homomorphic addition and multiplication.

**Step 3: Extend Functionality to Include Homomorphic Subtraction**

• Implement a function for homomorphic subtraction to showcase a broader range of arithmetic operations.

**Step 4: Implement Aggregation Operations**

• Extend your demo to support operations like calculating the sum or average of encrypted data, showcasing homomorphic encryption's usefulness for privacy-preserving data aggregation.

**Step 5: Frontend Development**

• Design a user-friendly interface that allows users to input values and view encrypted and decrypted results.

• Incorporate interactive charts or visualizations to enhance user understanding.

**Step 6: Implement Approximation Techniques**

• Explore and implement approximation techniques to balance accuracy and efficiency in your homomorphic encryption operations.

**Step 7: Security Analysis**

• Perform a basic security analysis of your homomorphic encryption implementation. Document the security guarantees it provides and highlight any limitations or considerations.