Mini-Project Specification Proposal

M.EIC025 - Computer Systems Security

Group 4:

Alexandre Chouzende - up202207429
Lucas Bessa - up202208396
Pedro João - up202204962

Theme: Fully Homomorphic Encryption: A Practical Lab on Privacy-Preserving Cloud Computation

1. Context and Motivation

As cloud computing and outsourced data processing become ubiquitous, maintaining the privacy of sensitive data is a critical security challenge. Traditional encryption requires data to be decrypted before computation, exposing it to the server. Fully Homomorphic Encryption (FHE) solves this by allowing computation directly on ciphertexts.

However, FHE is mathematically dense, and existing educational materials often lack approachable, hands-on exercises for modern FHE implementations. This project aims to bridge that gap. We propose an original lab exercise that moves away from the underlying lattice mathematics and focuses on the practical application and engineering constraints of Homomorphic Encryption using Python-based libraries.

2. Educational Objectives

By completing this laboratory, a typical Computer Engineering student will be able to:

- Understand the fundamental differences between Partial (PHE), Somewhat (SHE), and Fully Homomorphic Encryption (FHE).
- Experience the limitations of early homomorphic schemes (e.g., operation restrictions).
- Understand the concept of "cryptographic noise" introduced during ciphertext operations and how it limits computation depth.
- Implement a privacy-preserving client-server architecture using a modern FHE library.

3. Lab Scenario and Structure

The lab places the student developing a secure cloud service for an institution. The institution needs the cloud to calculate risk scores based on personal data without ever accessing the plaintext data.

The laboratory guide will be divided into three main practical tasks:

- **Task 0: The Privacy Flaw (Standard Encryption):** Students will begin by implementing the service using standard symmetric encryption (e.g., AES). They will observe that to compute the algorithm, the server must first decrypt the data, exposing the sensitive plaintext in the server's memory. This establishes the critical vulnerability that necessitates FHE.

- **Task 1: The PHE Limitation:** Students will start with a rudimentary script demonstrating Partial Homomorphic Encryption. They will successfully add encrypted values but will encounter an intentional failure when attempting to multiply/add them (depending on the cryptosystem), demonstrating the operational limits of PHE.

- **Task 2: SHE and the Noise Problem:** Students will transition to a modern library. They will be tasked with computing a complex polynomial (e.g., y = $3x^3 + 2x^2 + x$) representing some algorithm. Without proper parameter configuration, the continuous multiplications will increase the cryptographic noise, resulting in decrypted "garbage" data. This visually demonstrates the limitation of Somewhat Homomorphic Encryption.

- **Task 3: Achieving FHE and Parameter Tuning:** Students will fix the implementation from Task 2 by properly configuring the scaling factor and the polynomial modulus degree, effectively managing the noise budget to allow the computation to complete successfully.