CSEC 759 Lab 2: Homomorphic Encryption

Kyle Mullen
Dept. of Cybersecurity
Rochester Institute of Technology
Rochester, New York, USA
Email: kam3634@rit.edu

I. Introduction

The sharing of sensitive data using encrypted means, using a variety of protocols, strengths, and data types, is a fundamental element of the modern online ecosystem. The financial transactions, personally identifiable information, and health conditions that one uses and shares in digital spaces are done with the expectation that no party has the motivation, desire, or capability to take advantage of it for their own gain.

Homomorphic encryption, within this lab environment, is built on the principle that a calculation can be performed server-side on an encrypted piece of data and return an accurate result - which can then be decrypted client-side using asymmetric private keys. At no point does plain text information - in this case characteristics of one's heart functionality - become visible to anyone monitoring traffic **or calculating a result**. The creation of an algorithm that can make accurate decisions without true access to the data at hand is vital for the preservation of privacy against potential server-side information exposure.

Within the limited functionality of the lab environment, this type of encryption scheme sufficiently preserves the privacy of clients seeking a diagnostic prediction on their risk of heart disease. Any scaled implementation would have to highly account for the computational inefficiencies of homomorphic schemes, the length of the encrypted data being handled, and the complexity of a more extensive data set. As developments in these aspects continue, this demonstration shows why the principles continue to be worth pursuing regardless.

II. Code

The code in use can be found on GitHub: https://github.com/smoov22/homomorphic

A. Challenges of Implementation

Two primary hurdles presented themselves during the course of the development process. While the representation of client-side and server-side processes as separate scripts communicating over a socket would be the best visual match for the events occurring, I felt that would be an unnecessary and potentially breakable element of the process, which is worth overlooking to simulate and designate each element of the process in a single script instead for ease of use. The functionality is demonstrated by first making a prediction with plaintext data, then making that same prediction and outputting an identical result with encrypted data.

The second primary hurdle was an outage of the database intended for use, the UCI Heart Disease repository. This led to both barriers to access documentation as well as the prevention of any live testing. As such, this script (homomorphic.py) is accompanied by a similar operation (homomorphic-csv.py) which instead operates on a provided portion of the data in question. No other functionality was changed, and course staff are appreciated for quickly handling the issue.

III. Results

After a sample of the first five records is processed to confirm validity of the homomorphic process, the 300 records are processed both in plaintext and in encrypted format - at very different speeds. While the first CSV took 10.3 seconds on one run to produce, the corresponding encrypted CSV took 9.3 minutes. These can certainly be further optimized in scaled implementations, but the demonstration that both produce an identical results table is nonetheless valuable.