CSE/CEN 598 Hardware Security and Trust
Project 5: Homomorphic Encryption
Due Date: Nov. 30, 2023

Introduction and Background

This project demonstrates the basic concept of fully homomorphic encryption (FHE) using the addition and multiplication operations over encrypted data through two open-source FHE libraries: SEAL and OpenFHE. Both libraries demonstrate these operations by implementing two popular FHE schemes: Brakerski-Gentry-Vaikuntanathan (BGV) and Brakerski/Fan-Vercauteren.

OpenFHE

OpenFHE is an open-source homomorphic encryption library powered by Duality Technologies. It is currently the most popular FHE library. It provides implementations for five FHE schemes: BFV, BGV, CKKS, FHEW, and TFHE. It also includes multiparty extensions of FHE for threshold FHE and proxy Re-encryption. You can learn more about openFHE here.

SEAL

SEAL is an open-source homomorphic encryption library provided by Microsoft. The library implements the most popular FHE schemes currently used today: BGV, BFV, and CKKS. You can read more about Microsoft SEAL here.

Assignment - 100 points + 20 bonus points

For this assignment, you have been provided with four binaries located in *binaries* folder:

- SEAL BGV
- SEAL BFV
- OpenFHE BGV
- OpenFHE BFV

Each binary implements homomorphic addition and multiplication of encrypted data received as input. It outputs the run time and results of homomorphic operations carried out on the data received. The schemes and libraries used in each binary are part of its name, so it will be easy to distinguish them.

To run these binaries, you need to pass three command line arguments which will be used as *input1*, *input2* and *input3*:

```
./openFHE_BFV 1 1 2
```

The three arguments passed to the binary are encrypted and multiplication and addition operations are carried out on the data. The results are then converted to plaintext and displayed in terminal (Figure 1).

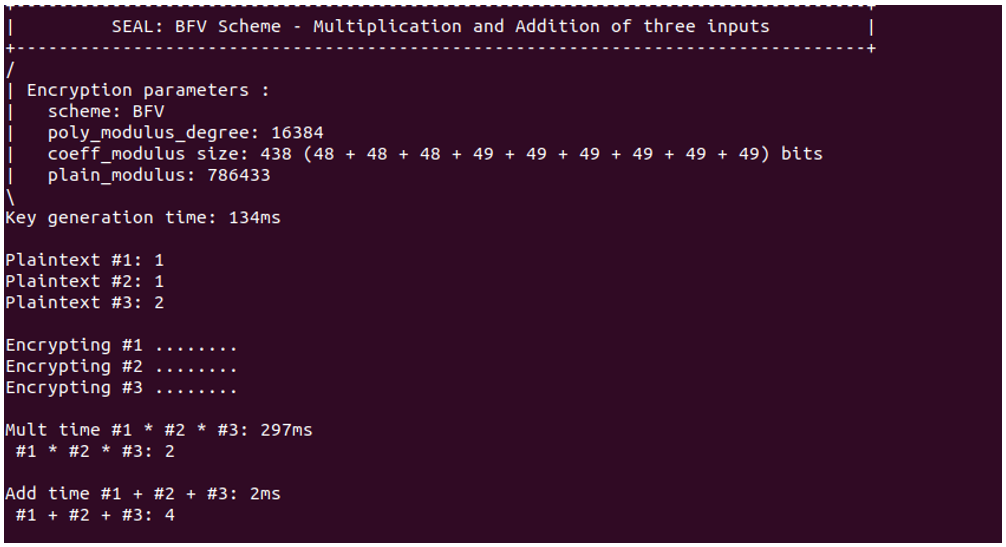

Figure 1: Sample output when running the provided binary.

Task 1.A

Generate at least ten unique combinations of inputs and pass them to the binaries. Show the table with runtimes of both addition and multiplication for each of the schemes and each of the libraries.

Task 1.B

Draw graphs to represent better the results generated in Task 1.A.

Task 1.C

Based on your collected results, which scheme do you think is better? Explain your findings in detail.

Task 1.D

Based on your collected results, which library has implemented the schemes better? Explain your findings in detail.

Task 1.E (Optional Bonus)

Write functions in C to perform identical multiplication and addition operations on 64-bit integer values. Time the execution of each and compare the runtimes to the FHE operations. What is the overhead for FHE execution? Note: You may need to execute your functions multiple times and take an average to get an accurate runtime.

Deliverables and Submission Requirements

Write a report answering the questions for each task. Be sure to address each question in the task. Save your report in the CSE598Project5Handout directory as a PDF. Zip your project5 directory with the file name **your_asurite_id**_project5.zip

Appendix I

This Appendix contains information about the build process for two libraries you will use for this assignment. The necessary binaries have already been compiled for you. These directions are provided so you can experiment independently with FHE implementations. Note: The build process may not work on the course VMs due to software incompatibility and resource (RAM) limitations.

Prerequisites

To run this project, you need to have the following dependencies:

- CMake
- G++
- GMP
- Clang

OpenFHE

Change the directory to *openfhe* and use the following instructions to setup the project:

```
mkdir build
cd build
cmake ..
make
make install
make testall (optional)
```

After changing the code, you can use the following command to compile any of the test programs.

```
make added_bgv
make added_bfv
```

To run the examples, you can run the binaries like this:

```
bin/examples/pke/added_bgv
bin/examples/pke/added_bfv
```

You can also create new files, which will be automatically linked during the make process.

SEAL

You can set up this library using the following instructions:

```
cmake -S . -B build -DSEAL_BUILD_EXAMPLES=ON
cmake --build build
sudo cmake --install build
```

To run the compiled work, you can locate the binary at *build/bin/sealexamples*.