Homomorphic Encryption Benchmarking with Microsoft SEAL

Contents

1 Introduction **1**

2 Features **1**

3 Project Structure **2**

4 Setup Instructions **2**
- 4.1 Prerequisites . . . 2
- 4.2 Steps . . . 2
  - 4.2.1 Clone and Build Microsoft SEAL . . . 2
  - 4.2.2 Add Benchmark Source Files . . . 3
  - 4.2.3 Build Benchmarks . . . 3
  - 4.2.4 Run Benchmarks . . . 3
  - 4.2.5 Visualize Results . . . 3

5 Technologies Used **4**

6 Author **4**

7 License **4**

1 Introduction

This project benchmarks the performance of Homomorphic Encryption (HE) operations using the Microsoft SEAL library. It evaluates execution time and memory usage for various operations and input types, with results logged to CSV files and visualized using Python scripts.

2 Features

- **Implementation**: Written in C++17 using the BFV scheme from Microsoft SEAL.
- **Operations Benchmarked**:
  - Ciphertext + Ciphertext
  - Ciphertext + Plaintext
  - Plaintext + Ciphertext
- **Input Types**:
  - Scalar (single integer) inputs
  - Vector inputs (sizes ranging from 1,000 to 10,000+)

- **Polynomial Modulus Degrees**: 1024, 2048, 4096, 8192, 16384
- **Metrics**:
  - Execution time (milliseconds)
  - Memory usage (kilobytes, measured using `task_info` on macOS)
- **Output**:
  - CSV logs for detailed analysis
  - Visualizations generated using Matplotlib and Seaborn

3 Project Structure

```
SEAL/
|-- native/
|   |-- examples/
|       |-- vector_benchmark.cpp      # Vector input benchmarking
|       |-- scalar_benchmark.cpp      # Scalar input benchmarking
|       |-- CMakeLists.txt            # CMake configuration for
   benchmarks
|-- build/
|   |-- bin/
|   |   |-- vector_benchmark          # Vector benchmark executable
|   |   |-- scalar_benchmark          # Scalar benchmark executable
|   |   |-- seal_benchmark.log.csv    # Log for vector benchmark
   results
|   |   |-- seal_scalar_benchmark.log.csv  # Log for scalar
   benchmark results
|   |   |-- vector_visualizer.py      # Python script for vector
   result visualization
|   |   |-- scalar_visualizer.py      # Python script for scalar
   result visualization
```

4 Setup Instructions

4.1 Prerequisites

- C++17-compatible compiler
- CMake
- Python 3.9+
- Git
- macOS (for memory profiling with `task_info`)

4.2 Steps

4.2.1 Clone and Build Microsoft SEAL

```
git clone https://github.com/microsoft/SEAL.git
cd SEAL
mkdir build && cd build
cmake .. -DSEAL_BUILD_EXAMPLES=ON
make
```

4.2.2 Add Benchmark Source Files

- Place `vector_benchmark.cpp` and `scalar_benchmark.cpp` in `SEAL/native/example`
- Update `SEAL/native/examples/CMakeLists.txt` to include:

```
add_executable(vector_benchmark vector_benchmark.cpp)
target_link_libraries(vector_benchmark SEAL::seal)
add_executable(scalar_benchmark scalar_benchmark.cpp)
target_link_libraries(scalar_benchmark SEAL::seal)
```

4.2.3 Build Benchmarks

```
cd SEAL/build
cmake ..
make
```

4.2.4 Run Benchmarks

- For vector inputs:

```
./bin/vector_benchmark
```

- For scalar inputs:

```
./bin/scalar_benchmark
```

- Output logs will be saved as:
  - `seal_benchmark.log.csv` (vector inputs)
  - `seal_scalar_benchmark.log.csv` (scalar inputs)

4.2.5 Visualize Results

- Create and activate a Python virtual environment:

```
cd SEAL/build
python3 -m venv venv
source venv/bin/activate
```

- Install required Python packages:

```
pip install pandas matplotlib seaborn
```

- Run visualization scripts:

```
python vector_visualizer.py
python scalar_visualizer.py
```

- Output images will be saved as:
  - `vector_benchmark_results.png`
  - `scalar_benchmark_results.png`

5 Technologies Used

- **C++17**: Core implementation of benchmarks
- **Microsoft SEAL**: BFV scheme for homomorphic encryption
- **Python 3.9+**: Visualization scripts
- **Pandas, Matplotlib, Seaborn**: Data analysis and visualization
- **macOS `mach/mach.h`**: Memory profiling

6 Author

BhShhan D., M.Tech Final Year
Part of Major Technical Project (MTP): Performance Evaluation of Homomorphic Encryption Libraries

7 License

This project is based on Microsoft SEAL, licensed under the MIT License.