Homomorphic Encryption

Student(s)

İlayda Begüm İzci
Şeyda Köse
Onat Kutlu

Faculty Member(s)

Erkay Savaş

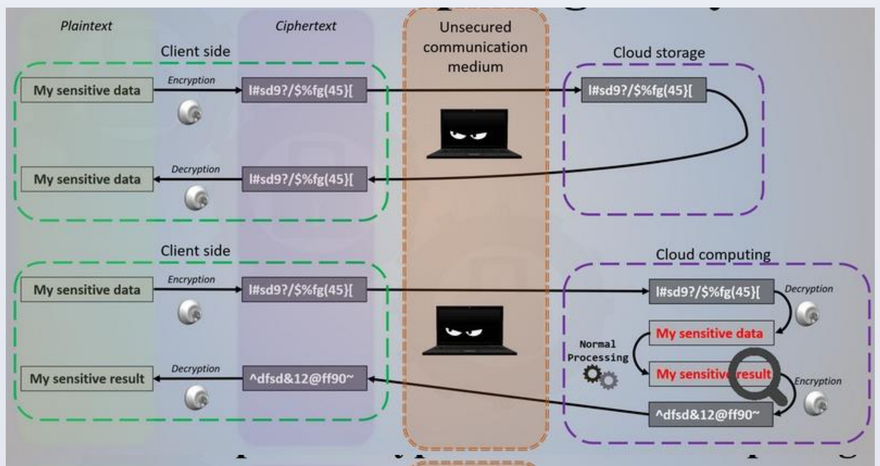
Introduction

With the modern communication technologies, privacy became one of the main concerns since the collected data may contain secret information. Homomorphic encryption allows to process encrypted data without decrypting it. Therefore, it is a powerful feature to enable computation and analytical processing on encrypted data by ensuring the confidentiality of that processed data. However, it was limited at first since it was supporting just one operation to manipulate the original plaintext by using only ciphertext.

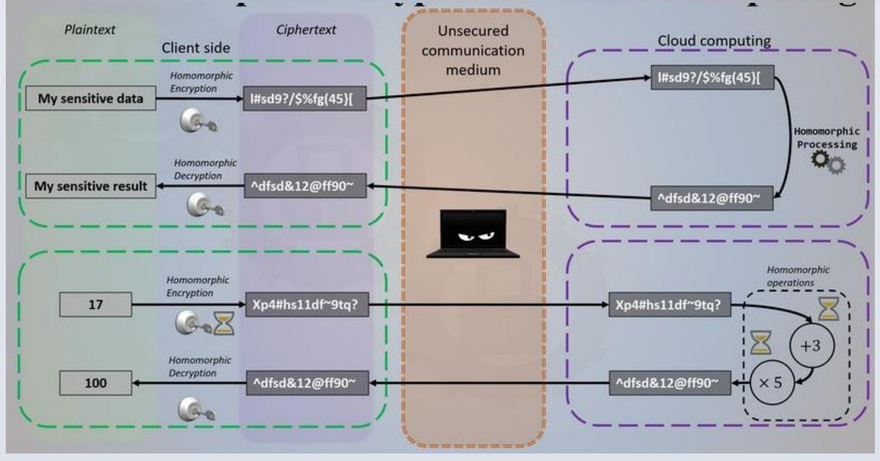
Fully Homomorphic Encryption (FHE) scheme which supports both addition and multiplication operations on ciphertexts is invented by Craig Gentry in 2009. This scheme permits more manipulation of the plaintext by modifying the ciphertext.

Throughout this project we aimed to introduce an efficient sorting algorithm that sort encrypted data with Fully Homomorphic Encryption scheme. We analyze several circuits and their multiplicative depth complexity. In this algorithm, we focused on the multiplicative depth of the comparison and sorting circuits to cope with massive noise growth. In our sorting scheme we used the homomorphic encryption library SEAL (Simple Encrypted Arithmetic Library).

Cloud Computing Today:



Homomorphic Encryption for Cloud Computing:



Objectives

- Understanding the concept of homomorphic encryption and its applications.
- Working on privacy-preserving data processing applications leveraging the power of homomorphic encryption and software libraries that implement it.
- Using these newly gained knowledge to write a sorting algorithm for encrypted integers.

Project Details

Algorithm 1: Compares two integers and constructs the comparison matrix

```
1: vector<vector<Ciphertext>> matrix(4, vector<Ciphertext>(4))   #Ciphertext matrix
2:for (int i=0; i<4; i++)                                        #Fills the matrix
3:      M[i][i] ← 0
4:          for (int j = i + 1; j<4 ;j++)
5:              M[i][j] ← Compare(X[i];X[j])*
6:              M[j][i] ← 1-M[j][i]
7:          end for
8: end for
```

Algorithm 2: Uses full adder to find sum of the rows

```
1: vector<vector<Ciphertext>> Fulladdermatrix(4, vector<Ciphertext>(2));
2: for (int k = 0; k < 4; k++)
3:          for (int l = 0; l < 4; l++)
4:                Sum← matrix[l][k]
5:          end for
6:         Carry← matrix[0][k] XOR matrix[1][k] XOR matrix[2][k]  XOR matrix[3][k]
7:         Fulladdermatrix[k][0] ← Carry
8:         Fulladdermatrix[k][1] ← Sum
9:end for              #Fulladder is the matrix holding encrypted bits of sum of the rows
```

Algorithm 3: Finds the largest integer

```
1: A←Fulladdermatris[0][0] * Fulladdermatris[0][1]*num1
2: B←Fulladdermatris[1][0] * Fulladdermatris[1][1]*num2
3: C←Fulladdermatris[2][0] * Fulladdermatris[2][1]*num3
4: D←Fulladdermatris[3][0] * Fulladdermatris[3][1]*num4
5: A+B+C+D
```

```
****************************************
********** Sorting Algorithm **********
****************************************

/ Encryption parameters:
| poly_modulus: 1x^16384 + 1
| coeff_modulus size: 438 bits
| plain_modulus: 1024
\ noise_standard_deviation: 3.19

Generating 4 random integers
integer1 :9
integer2 :3
integer3 :14
integer4 :4
Comparing numbers and generating comparison matrix
Average comparison time [1295684 microseconds]
Comparison matrix has been generated [18419200 microseconds]
Decrypted version of comparison matrix
[ 0 ][ 0 ][ 1 ][ 0 ]
[ 1 ][ 0 ][ 1 ][ 1 ]
[ 0 ][ 0 ][ 0 ][ 0 ]
[ 1 ][ 0 ][ 1 ][ 0 ]
Average time for calculating sum [1901598 microseconds]
Average time for calculating carry [871407 microseconds]
 fulladder matrix generated [7862127 microseconds]
 Decrypted version of fulladder
[ 1 ][ 0 ]
[ 0 ][ 0 ]
[ 1 ][ 1 ]
[ 0 ][ 1 ]
There is7 noise budget left
Highest is14
Highest found [8149383 microseconds]
Total time [37371819 microseconds]
```

*** Compare** function compares two encrypted ℓ-bit integers E(X) and E(Y) such that x_i is the i^{th} bit of X and outputs 1 or 0.

Less Than Circuit $E(x_k) < E(y_k) \rightarrow (y_k \cdot (x_k \oplus 1))$
Equality Circuit $E(x_t) = E(y_t) \rightarrow (x_t \oplus y_t \oplus 1)$

Compare(X,Y) $\rightarrow E(X) < E(Y) = \sum_{k=1}^{l} (E(x_k) < E(y_k) \prod_{k<t<l} (E(x_t) = E(y_t))$

Conclusion and Future Work

In conclusion our algorithm is successful to find the integer in desired position (ex: the largest integer, the smallest integer etc.). In the table below you can see the time elapsed to complete tasks in seconds. However we couldn't manage to sort all of the integers in a row. In the course of the project, our main problem was reducing the noise of ciphertexts. Most of the operations consumed a lot from our initial noise budget thus it was not possible to perform as much operations as we needed to compare and sort the numbers. To overcome this problem, we tried to optimize our algorithm by decreasing the number of operations and setting parameters effectively. As a future task, we are planning to use packing techniques such as CRT(Chinese Remainder Theorem) batching to do parallel operations. By this way we expect to get lower depth, resulting in an accelerated process and reduced noise.

Time Scale	
Average comparison time for 2 encrypted integers	1,3
Time for generating the comparison matrix	18
Average time for calculating sum	1,9
Average time for calculating carry	0,8
Time for generating fulladder matrix	7,8
Time past for finding the highest number	8,1
Total process time	37.3 seconds

REFERENCES

- Çetin, G. S., Doröz, Y., Sunar, B., & Savaş, E. (2015). Depth Optimized Efficient Homomorphic Sorting. *Progress in Cryptology -- LATINCRYPT 2015*, 61-80.
- Is there any good tutorial/resource to understand Homomorphic Encryption from scratch? OR any flow of background study to understand it? - Quora. (n.d.). Retrieved from https://www.quora.com/Is-there-any-good-tutorial-resource-to-understand-Homomorphic-Encryption-from-scratch-OR-any-flow-of-background-study-to-understand-it
- Gentry, C.: A Fully Homomorphic Encryption Scheme. Ph.D. thesis, Stanford University (2009)