Term Project: Developing a privacy-preserving application using homomorphic encryption.

Homomorphic encryption (HE) enables computations on data in an encrypted state. For example, if the encrypted values of random numbers a and b are denoted as E(a) and E(b), (E(): Encryption algorithm), HE can perform multiplication and addition on ciphertext, which means there are special functions Add() and Mult() such that Add(E(a),E(b)) → E(a+b), Mult(E(a),E(b)) → E(a \*b). Thus, by decrypting the result, we can obtain the result of multiplication and addition. Since we can perform such calculations over the hidden plaintext in the ciphertext WITHOUT decrypting them, we say it's an HE. Moreover, if it's possible to compute such operations over infinite number of times. We say the HE as a fully HE.

There are various fully HE algorithms, but we use the CKKS (Cheon-Kim-Kim-Song) algorithm, which is widely used in the field of machine learning and data mining. The CKKS algorithm can operate on the ciphertext of complex numbers, and operates correctly when their magnitudes are at most 1. Within this range of input values, CKKS provides addition and multiplication functions over encrypted input. Expanding this, it is possible to perform homomorphic polynomial operations. That is, if a value x is in the encrypted ciphertext, we can compute an arbitrary polynomial function f(x)=a\_(n-1) x^(n-1)+a\_(n-2) x^(n-2) )+⋯+a\_1 x^ +a\_0 on the encrypted input x without decrypting the input.

In addition, CKKS homomorphic encryption provides SIMD (Single Instruction Multiple Data) function. In other words, it is possible to encrypt tens of thousands of numbers at one an put them in a single ciphertext as if multiple elements can be put in an array, and simultaneous operation (Vector computaton) is possible using the operations provided by HE. Therefore, polynomials for multiple encrypted values can be calculated with only the amount of computation required to perform a single polynomial. For example, for two vectors consisting of five elements as shown below, element-wise multiplication can be completed by a single homomorphic multiplication between two ciphertext.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

(Enc) --> (Ciphertext: values hidden)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

X (Homomorphic multiplication)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.2 | 0.2 | 0.3 | 0.3 | 0.4 |

(Enc) --> (Ciphertext: values hidden)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.02 | 0.04 | 0.09 | 0.12 | 0.2 |

↓

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

<-- (Dec) (Result of multiplication)

(Multiplication result: plaintext)

This procedure can be implemented as below box.

import piheaan as heaan

import os

# set parameter

params = heaan.ParameterPreset.FGb

context = heaan.make\_context(params) # context has parameter information

heaan.make\_bootstrappable(context) # Make parameter bootstrapable

key\_file\_path = "./keys"

# create secret\_keys

sk = heaan.SecretKey(context) # create secret key

os.makedirs(key\_file\_path, mode=0o775, exist\_ok=True)

# create public\_key

key\_generator = heaan.KeyGenerator(context, sk)

key\_generator.gen\_common\_keys()

eval = heaan.HomEvaluator(context,pk) # to load pi-heaan math

dec = heaan.Decryptor(context) # for decrypt

enc = heaan.Encryptor(context) # for encrypt

log\_sots=15

# if log\_slots = 15 🡪 the number of slots per ciphertext is 2\*\*15

# log\_slots is used for the number of slots per ciphertext

# log\_slots value depends on the parameter used. (ParameterPreset)

# 15 is the value for maximum number of slots, but you can also use a smaller number. (ex. 3, 5)

# generate Message1

1. Create a list with values of your choice within the number of slots.

2. Declare a message.

3. Assign the values of the list to the message slots one by one.

m1\_ = [0.1, 0.2, 0.3, 0.4, 0.5]

m1 = heaan.Message(log\_sots)

for i in range(len(m1\_)):

m1[i] = m1\_[i]

# encrypt m1 to ctxt1 using public\_key

ctxt1 = heaan.Ciphertext(context)

enc.encrypt(m1, pk, ctxt1)

# generate Message2

m2\_ = [0.2, 0.2, 0.3, 0.3, 0.4]

m2 = heaan.Message(logN-1)

for i in range(len(m2\_)):

m2[i] = m2\_[i]

# encrypt using public\_key

ctxt2 = heaan.Ciphertext(context)

enc.encrypt(m2, pk, ctxt2)

# calculate Mult : result ← ctxt1 \* ctxt2

result = heaan.Ciphertext(context)

eval.mult(ctxt1, ctxt2, result)

# decrypt result to result\_m using secret\_key

result\_m = heaan.Message(logN-1)

dec.decrypt(result, sk, result\_m)

print(result\_m)

To execute the above code, the pi-heaan module must be installed, and refer to https://pypi.org/project/pi-heaan/. This module emulates HE and does not actually perform the homomorphic operations. However, it has the advantage of being able to understand HE, easily implement and test the applications utilizing the CKKS HE.

Tasks:

1) Research on HE and its application areas: You should conduct research on HE through various means such as internet search and investigate its related applications. The findings from your research should be added to the proposal report.

2) Run the pi-heaan code described above and familiarize yourself with its usage through the manual.To create an application using HE, you need to be familiar with the usage of the pi-heaan code. Run the code described above and the code from the pi-heaan website, and try various applications.

3) Develop a privacy-preserving application using HE with pi-heaan (Python environment). You will work in a team of three to develope a privacy-preserving application using HE. The application you create will be evaluated based on the following criteria:

1) Creativity: The privacy application you create should have unique improvements compared to previously proposed applications.

2) Utility: The privacy application you create should solve existing problems in privacy/security-related applications and have value for people to use.

3) Difficulty: The privacy application you create should be designed and developed based on the creative technology you propose. It will require a lot of effort to create.

4) Comprehension: The application and its source code you create should be easily understandable with the help of comments and reports, and be easily extendable based on them.

In your implementation, the following should be noted:

1) Encoding method of input data: When processing encrypted data, there will be cases where the input information needs to be homomorphically encrypted and then processed in an encrypted state. In this case, the input data needs to be encoded for efficient processing. You must clearly explain the encoding method for plaintext data before encryption.

2) Data processing algorithm for encrypted data: To process data in the homomorphically encrypted state, a processing algorithm needs to be implemented using homomorphic operations (such as addition and multiplication). You need to describe this method in detail.

3) Explanation of the reason for providing privacy preservation: In your implementation, there will be parts where data is processed before or after encryption/decryption. Based on this analysis, you need to explain why and how privacy or security is supported to certain data in the proposed application.

Deliverables

\* Proposal

1) Survey report: research contents related to HE and its applications

2) Proposal: the introduction and an abstracted description on how to implement the proposed application.

\* Progress report

- Mid-term report on the implementation status, the execution method of the implementated part, and the method to verify the results

\* Final report (or presentation)

1) Introduction of the developed application (final version) + plan and changes comparing to the proposal + survey on the related work

**2) Implementation code (attached separately from the report) (should include comments)**

3) Program design and structure report

- The following sections must be included:

a. Design overview (program structure and explanation in the macro scope)

b. Detailed design (design contents for each component in a.)

c. Explanation of key code (linked to the detailed design in b.)

d. Program execution method (required packages, usage, etc.)

다음명령어를실행 **pip install pillow pyzbar json base64 piheaan tempfile qrcode urllib3 numpy**

e. Key performance evaluation (execution time, accuracy, etc.)

**execution time: within 0.8sec, accuracy: perfectly well-generated**

\* Provided materials

- Privacy-preserving machine learning example code (logistic regression)

- An extended version of pi-heaan manual

\* Schedule

1. Project material released (4th week)

2. Proposal due (6th week)

3. Progress report (9th week)

4. Final report and implementation result (14th week)

\* Deliverable

1. Proposal: A ppt file that explains an overview of your implemenation. It must include the motivation/creativeness/technical challenges

2. Progress: A ppt file that shows the progress of your implementation and your implementation that produces a partial result.

3. Final: A ppt file that shows the full explanation of your implementation. The source code of your implementation including detailed comments. You may demonstrate your implemenation with presentation with the submitted ppt file.

\* Evaluation

- 15% of total points is assigned to this project

- Proposal (3%), Progress (3%), Final (9%)

\* How to use Jupyter Notebook

1. install Anaconda

* Download the appropriate version based on your computer environment.
* <https://www.anaconda.com/products/individual>

2. After installation, you can see the Anaconda3 folder.

텍스트이(가) 표시된 사진

자동 생성된 설명

3. Click ‘Jupyter Notebook (anaconda3)’.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명Then you can check this window.

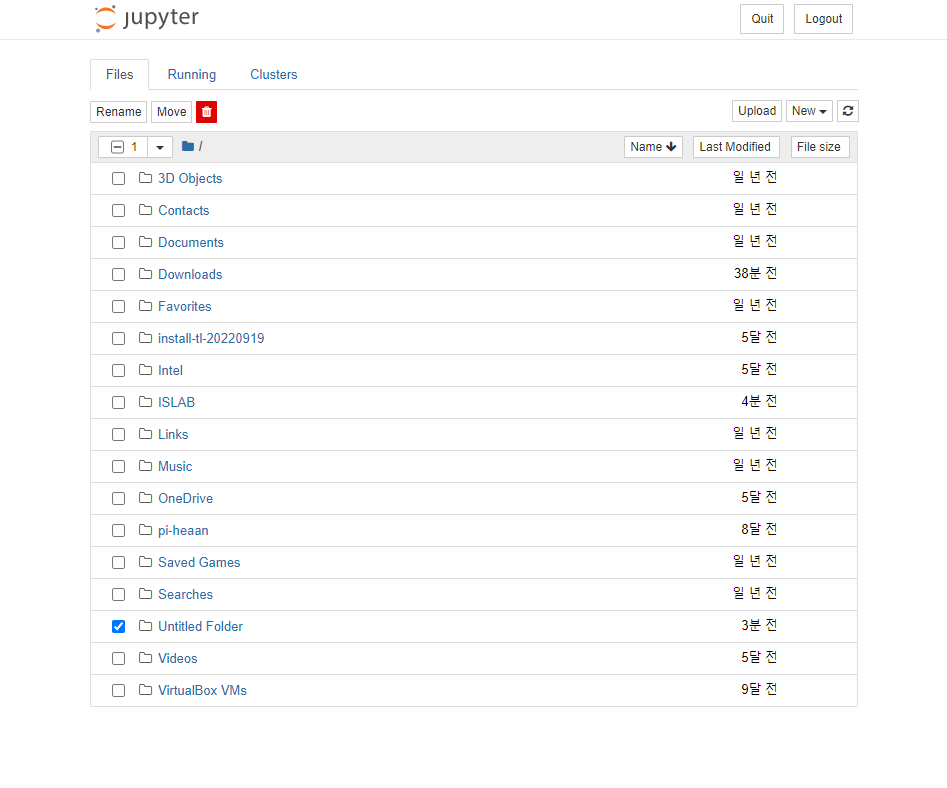
4. You can also see this window.

* ➀ make new file or folder
* ➁ select folder(or file) and you can rename, move, remove it

테이블이(가) 표시된 사진

자동 생성된 설명

➀



➁

5. Let’s upload practice file.

* Go to the directory where you want to work.
* Drag and drop the file you want to use.

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

* Then click the ‘Upload’ button.
* Now you are ready to practice

Or you can use <https://jupyter.org/try-jupyter/lab/> if you want to enjoy the Jupytor on your web browser.

“If you encounter any difficulties using Piheaan, don't hesitate to request assistance from a TA”