Fully Homomorphic Encryption

EE793: Course Project Report

Archit Gupta 2101000021

Mayank Gupta 210101002

Soham Inamdar 210100149

Swadhin Dash 210020142

May 16, 2024

Project Statement

Demonstration of fully homomorphic encryption, showing robustness over addition and multiplication for multiple message subspaces. Logistic regression was also implemented to show the effects. Primary paper referred was "Homomorphic encryption", Shai Halevi (IBM Research).

Introduction

Homomorphic encryption is a form of encryption that allows computations to be performed on encrypted data without first having to decrypt it. As the data size increases, we tend to use cloud services for storage and computation purposes. However, due to security reasons, we can't store the data directly and hence, we need some encryption scheme which will not affect the accuracy of computations performed. Most mathematical operations can be decomposed into a sequence of multiplication and addition operations, so it is sufficient to check for homomorphism over these two.

Central Idea

The encryption of messages was tested over addition and multiplication for homomorphism. Two messages, a and b were to be encrypted. Let c_a and c_b be the corresponding ciphers.

For additivity, we have demonstrated that:

$$a + b = Dec(Enc(a) + Enc(b))$$

For multiplication,

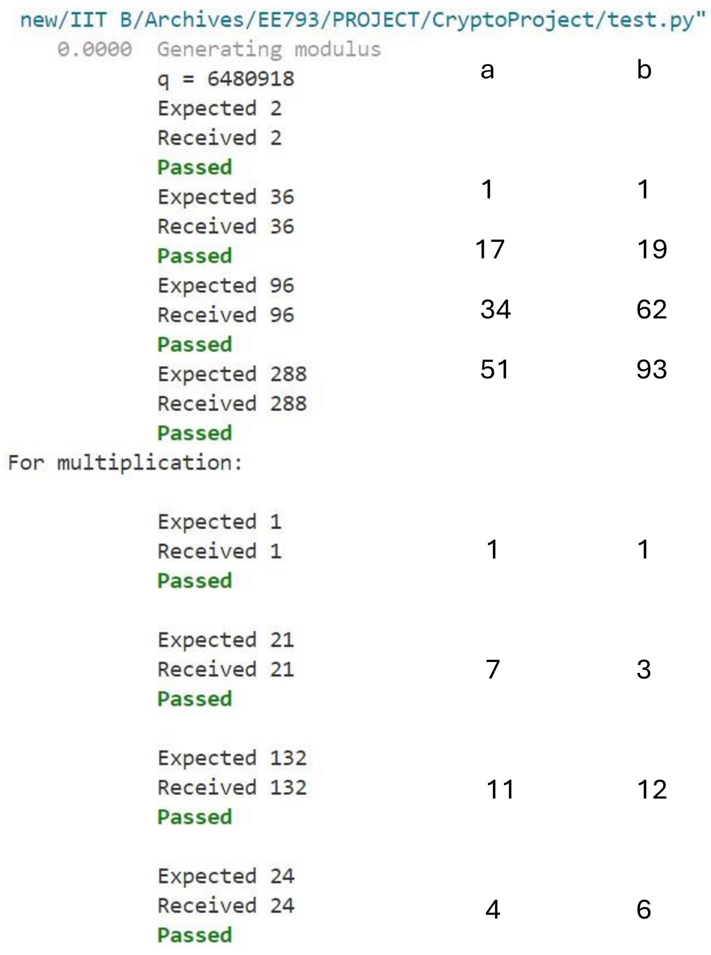
$$a * b = Dec(Enc(a) * Enc(b))$$

Implementation

1. We have implemented homomorphic encryption from scratch and verified it for multiplication and addition as explained above.
2. Apart from this, in a separate ipynb file we trained a logistic regression model without any encryption and tested it on plaintext as well as encrypted data. Then, we trained the model on encrypted data and tested it. We get similar values of accuracy in all cases.

Results

On running the module for a multiple test cases, we obtained the following results:



```
 new/IIT B/Archives/EE793/PROJECT/CryptoProject/test.py"
    0.0000  Generating modulus
            q = 6480918            a           b
            Expected 2
            Received 2
            Passed
            Expected 36            1           1
            Received 36
            Passed                17          19
            Expected 96
            Received 96            34          62
            Passed
            Expected 288           51          93
            Received 288
            Passed
For multiplication:

            Expected 1
            Received 1              1           1
            Passed

            Expected 21
            Received 21             7           3
            Passed

            Expected 132
            Received 132           11          12
            Passed

            Expected 24
            Received 24             4           6
            Passed
```

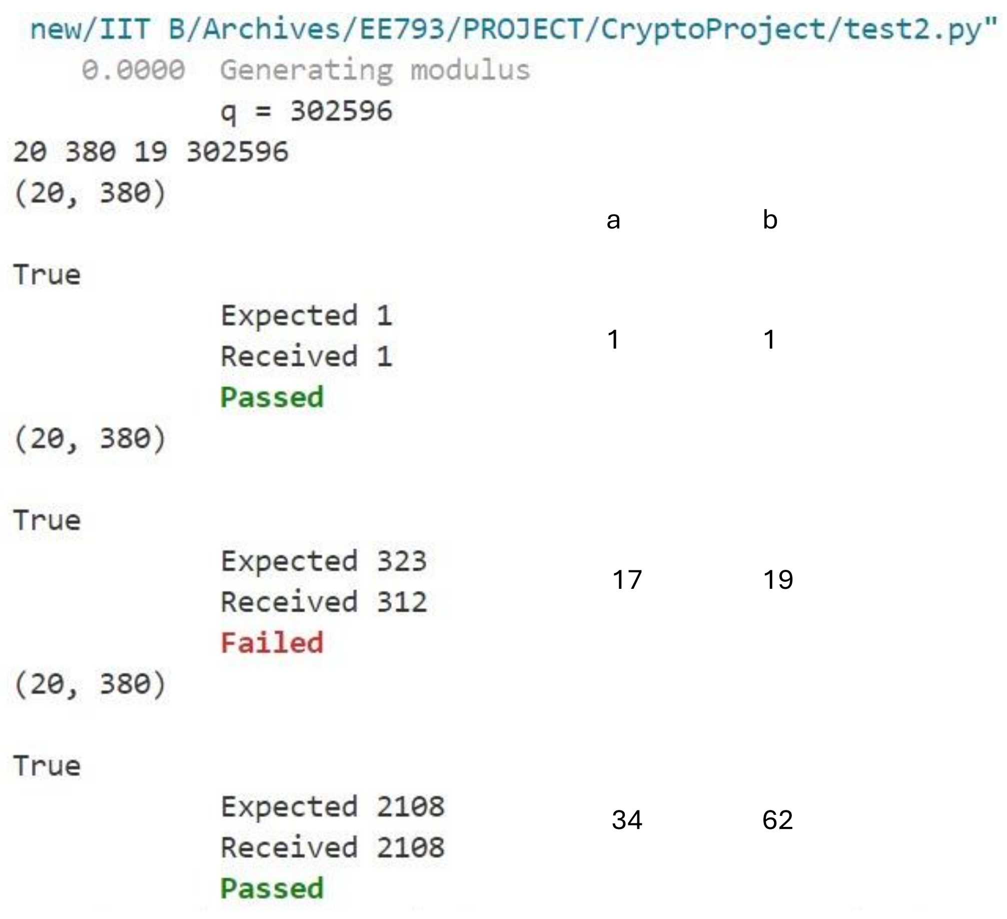
The multiplication results are dependent on an error threshold, as described in the paper by:

We can then define multiplication as $\mathbf{C}^{\times} = \mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2)$. This will be equal to:

$$\mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2) = \underbrace{\begin{pmatrix} -\mathbf{A}_1 \mathbf{G}^{-1}(\mathbf{C}_2) \\ \mathbf{s}^\top \mathbf{A}_1 \mathbf{G}^{-1}(\mathbf{C}_2) + \mathbf{e}_1^\top \mathbf{G}^{-1}(\mathbf{C}_2) \end{pmatrix} + \mu_1 \begin{pmatrix} -\mathbf{A}_2 \\ \mathbf{s}^\top \mathbf{A}_2 + \mathbf{e}_2^\top \end{pmatrix}}_{\text{Error}} + \mu_1 \mu_2 \mathbf{G}$$

As the error increases with the increase in the value of the message, this was also evident in our results.

On increasing the modulus of the message.



```
 new/IIT B/Archives/EE793/PROJECT/CryptoProject/test2.py"
    0.0000  Generating modulus
            q = 302596
20 380 19 302596
(20, 380)

True
            Expected 1
            Received 1
            Passed
(20, 380)

True
            Expected 323
            Received 312
            Failed
(20, 380)

True
            Expected 2108
            Received 2108
            Passed
```

a	b
1	1
17	19
34	62

Logistic Regression Logs

Accuracy on Plain Text Data = 0.982421875

Encryption of the test-set took 2 seconds

Evaluated test_set of 512 entries in 1 seconds

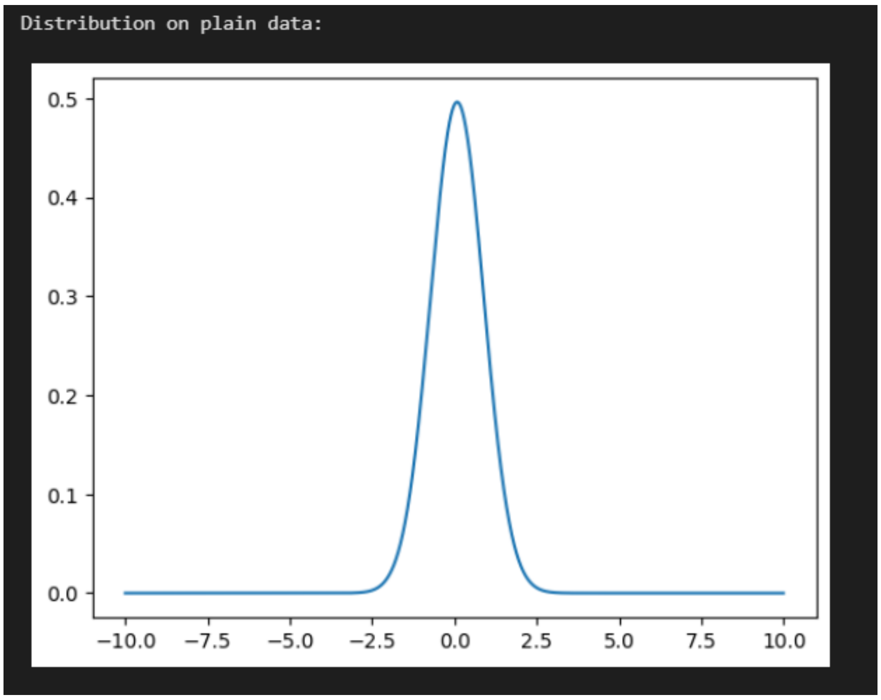
Accuracy: 495/512 = 0.966796875

Difference between plain and encrypted accuracies: 0.015625

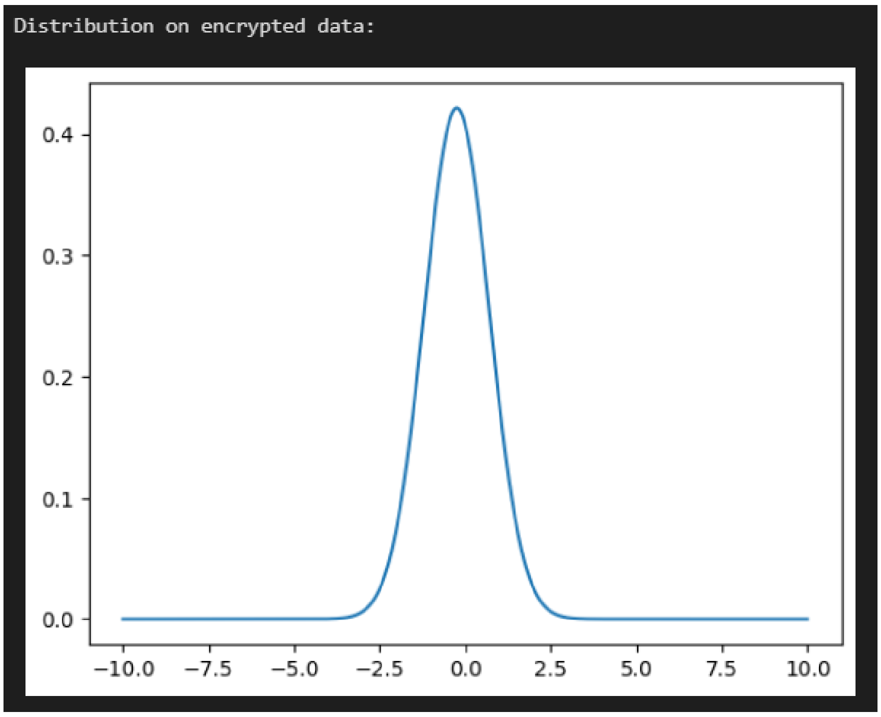
(all the code can be found in code.ipynb file)

Training an Encrypted Logistic Regression Model on Encrypted Data

We study the distribution of x.dot(weight) + bias in both plain and encrypted domains. Making sure that it falls into the range [-5,5], which is where our sigmoid approximation is good at. To implement this, we will use the help of regularization.



```
Distribution on plain data:
```


```
Distribution on encrypted data:
```

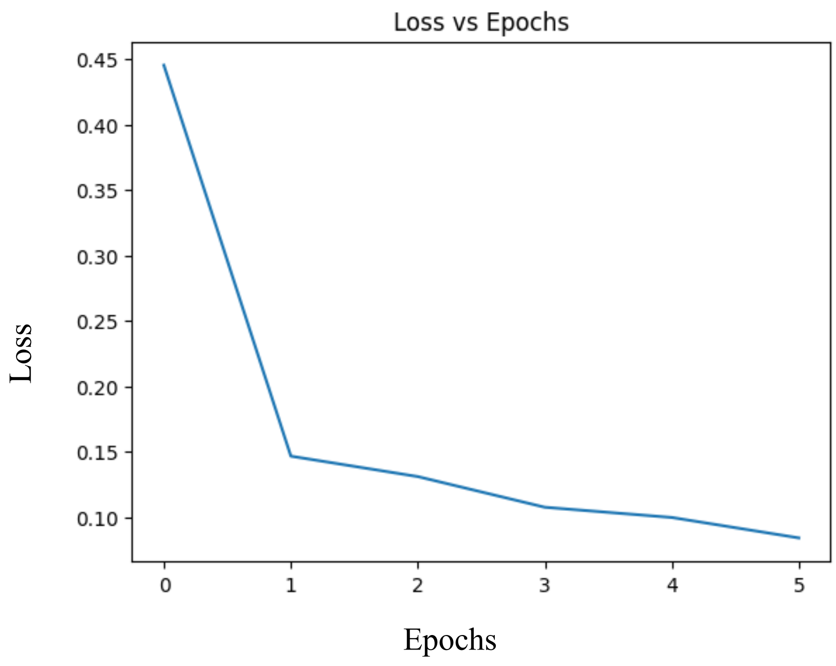
Most of the data falls in the [-5, 5] region validating our approximation.

Proceeding with Training -

Average time per epoch: 115 seconds

Final accuracy is 0.916015625

Difference between plain and encrypted accuracies: 0.06640625



Loss vs Epochs for model trained on encrypted data

We have successfully implemented the paper with satisfactory results and conclusions. Paper [2] gave us insights how multiplication can be implemented and how it's innately different from addition. The lectures have immensely helped us to successfully complete this project.

References -

1. Homomorphic Encryption, Shai Halevi (IBM Research), April 2017
2. Michele Minelli. Fully homomorphic encryption for machine learning. Cryptography and Security [cs.CR]. Université Paris sciences et lettres, 2018. English. ffNNT : 2018PSLEE056ff. fftel-01918263v2f