"Homomorphic Encryption Scheme for Privacy-Preserving Computation on Encrypted Data"

Dr. Jyoti Shetty[1], Shreyas Krishnaswamy[2], Sravani H[3], Shreehari G Bhat[4], Vidwath H Hosur[5]

jyotis@rvce.edu.in[1] , shreyask.cs22@rvce.ed.in[2], sravanih.cs22@rvce.edu.in[3], shreeharigbhat.cs22@rvce.edu.in [4], vidwathhhosur.cs22@rvce.edu.in[5]

[1]Assistant Professor, Department of Computer Science and Engineering, R V College of Engineering

[2,3,4,5]BE students, Department of Computer Science and Engineering, R V College of Engineering

ABSTRACT

Arising concerns about information privacy and security,homomorphic encryption has become one of the most powerful tools to execute any computation on encrypted data without decrypting it. We present an effective implementation of the homomorphic encryption scheme to enable machine learning models to work with encrypted data sets directly and hence provide privacy for the data across the whole analytics pipeline.

Our solution discusses some important challenges: computational efficiency, compatibility of models, and limitations imposed by HE schemes. We provide an in-depth discussion of trade-offs of privacy, performance, and aspects of practical implementation. As a part of the effort to demonstrate how HE can be used in concrete applications, this paper represents one step toward enabling privacy-preserving machine learning as well as one direction toward scalable data analysis in a setting sensitive for privacy. Our results contribute to the fast growing literature on combining cryptographic techniques with machine learning by pushing the limits regarding the scale of secure data processing.

1. INTRODUCTION

Homomorphic encryption has become a very strong tool in the area of data privacy, finding its applications across many areas, including machine learning. Classic machine learning applications rely on direct access to raw data, which poses significant privacy risks, especially when sensitive information comes into consideration. Homomorphic encryption enables the performance of computations on encrypted data without leaking any information regarding the underlying data. It also allows a computation on data securely, without its owner revealing the data to the service provider, which solves concerns regarding privacy.

Working with the Ridge Regression model in the machine learning framework involves several steps to be done in order to protect customers' privacy. First, the customer encrypts their data under a homomorphic encryption scheme, for example, the Paillier system. This encrypted data is subsequently sent to the machine learning provider, who will carry out all the computations on this encrypted data using a pre-trained model. Because of this, the confidentiality of the raw data is always maintained, as the provider does not access it. Computed results are encrypted and passed to the customer, who decrypts them using his private key. This approach guarantees security for sensitive customers' data throughout the whole machine learning process, including interpretation of the results.

1.1. PROBLEM STATEMENT

While this reflects an increasing reliance on cloud-hosted AI/ML models in computing sensitive data, this situation has ripened the concerns of data privacy and security. Although encryption is a standard approach to protect data both during storage and transmission, traditional methods require decryption prior to any computational processing. This step in decryption by nature exposes the data within the cloud to potential vulnerabilities, whereby sensitive information may be accessed by unauthorized entities, which could be malicious actors or even the cloud service providers themselves. This is a serious challenge in maintaining confidentiality and integrity of sensitive information, if third-party cloud services are utilized. After all, this puts an exceptionally high demand on having a solution that keeps the data encrypted throughput computation for organizations in various fields such as healthcare, finance, and government using these cloud-hosted AI/ML models. This is all the more important, considering the sensitivity of the data being processed by this class of models; any breach in confidentiality might prove disastrous.

1.2. MOTIVATION

This work is motivated by the increasingly critical need to protect sensitive data throughout the computational lifecycle, especially as organizations continue to adopt cloud-hosted AI/ML models for processing large amounts of personal, financial, and governmental information. Traditional encryption methods that provide a high level of assurance for data at rest and during transmission fall short when data must be decrypted for processing by AI/ML models. The decryption process for this introduces a number of vulnerabilities, as it makes the data susceptible to breaches in the cloud, where unauthorized access by malicious entities or even the cloud service providers themselves is a genuine concern.

1.3. OVERVIEW OF SYSTEM:

The system provides secure machine learning on encrypted data using homomorphic encryption, in particular, the Paillier system, for the protection of sensitive customer data. First of all, the customer generates a private/public key pair. Then, the customer encrypts the data with the generated public key. Then the encrypted data is sent to the machine learning service provider, which applies a pre-trained Ridge Regression model to the data without access to the information in unencrypted form. Computation is performed on the ciphertext data, returning results in encrypted form. The client can then decrypt such results with his private key. This ensures that even sensitive data remains encrypted during all phases of processing, and is never revealed to the service provider.

2. HOMOMORPHIC ENCRYPTION-PAILLIER ALGORITHM

2.1.PAILLIER CRYPTOSYSTEM

Basically, Paillier cryptosystem is one form of public-key encryption algorithm designed to support homomorphic operations, important for preserving privacy.

One of the salient features of the Paillier cryptosystem is to natively support homomorphic addition, meaning that addition of encrypted values can be performed directly on such values without the need for their decryption. More precisely, given two encrypted values, it is possible to homomorphically sum them in order to obtain an encryption of the summation of the two original plaintext values. This property renders Pailler particularly useful in applications that require computations on private data in such a manner that confidentiality is maintained.

In practice, for example in machine learning, the Paillier cryptosystem provides a means of carrying out computations on private data in a secure way by permitting operations to be conducted on encrypted inputs

2.1.1.MATHEMATICAL FOUNDATION OF THE PAILLIER CRYPTOSYSTEM

The basis of the cryptographic algorithm of Paillamper is homomorphic encryption. Homomorphic encryption provides processing over encrypted data by exploiting the structure of modular arithmetic and hardness of some number-theoretic problems. More specifically, the following ideas describe the very core of this encryption:
Homomorphic Addition: For a pair of encrypted values, their sum can be computed directly in the encrypted domain, i.e., without decrypting the values. This is done because of the property that the encryption of a sum is equal to the product of the encryptions.

The system operates with the key pair-one public key for encryption and another private key for decryption. Its security relies on the hardness of factoring big numbers and solving discrete logarithm problems in modular arithmetic.

```
31
32    def encrypt(message, public_key):  3 usages (1 dynamic)
33
34        n, g = public_key
35        r = np.random.randint( low: 1, n)
36        n2 = n * n
37        c = (mod_exp(g, message, n2) * mod_exp(r, n, n2)) % n2
38        return c
39
40    def decrypt(ciphertext, private_key, public_key):  1 usage
41        n, _ = public_key
42        lambda_n, mu = private_key
43        n2 = n * n
44        l = (mod_exp(ciphertext, lambda_n, n2) - 1) // n
45        message = (l * mu) % n
46        return message
47
```

2.1.2 ADVANTAGES OF PAILLIER CRYPTOSYSTEM

Preserving Privacy: It keeps sensitive data private during computation because only the encrypted data is processed. Homomorphic capabilities: it allows secure computations on encrypted data using homomorphic addition, which is very important in different scenarios where analytics should be preserving privacy.

2.2. PAILLIER CRYPTOSYSTEM IN MACHINE LEARNING

Paillier cryptosystem allows taking into account some machine learning techniques such as **Ridge Regression**: Secure computation: the training and evaluation of machine learning models on encrypted data protecting the data privacy against unauthorized access.
Model Evaluation in the Encrypted Form: Now, after the training phase, the model returns an encrypted result to the customer who decrypts the result using his private key.

2.2.1. KEY FEATURES OF PAILLIER IN ML

Encryption and decryption: Generally, public keys are used in encrypting the input data and the customer uses his own private key to decrypt the results such that sensitive information is kept private during computation.
Data Privacy: The Pailler cryptosystem allows computations on encrypted data, thereby keeping the privacy of customers' data safe from the machine learning service provider.

2.2.2. ADVANTAGES IN THE CONTEXT OF ML

Improved Security: It offers a very viable way to perform machine learning on sensitive information while keeping privacy intact.

Compatibility with Encrypted Data: This allows one to embed the privacy-preserving machine learning technique in various practical settings where data confidentiality is of primary importance.

2.3 IMPROVING EFFICIENCY BY USING OPTIMAL ALGORITHMS

```
def sieve_of_eratosthenes(limit):  1 usage

    primes = []
    sieve = [True] * (limit + 1)
    sieve[0] = sieve[1] = False

    for start in range(2, limit + 1):
        if sieve[start]:
            primes.append(start)
            for multiple in range(start * start, limit + 1, start):
                sieve[multiple] = False

    return primes
```

3. TRAINING ML MODELS TO PROCESS HOMOMORPHICALLY ENCRYPTED DATA

3.1. DATASET PREPARATION

Thus, each model is trained by using a dataset prepared for the specific task at hand. In any case, such a dataset has to be pre-prepared by a number of preprocessing steps in such a way that it suits the input format of the model. It normally includes tokenization, padding, and possible sequence truncation to meet the model's limitations concerning input size.
This is a dataset retrieved from Kaggle, a data science community, and imported into our code for further analysis.

```
df = pd.read_csv('employee_data.csv')
y = df.salary
X = df.drop('salary', axis=1)
scaler = preprocessing.StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=0)
```

3.2. TRAINING MODEL

First, the process of training starts with the initialization of a Ridge Regression model. This model is specialized in order to handle overfitting with the concept of regularization. Ridge Regression handles this problem by adding a regularization term to the loss-a parameter that is often called alpha. The example uses an alpha value of 1.0 because this balances out the trade-off between fitting the training data and simplicity of the model to avoid overfitting.

Then, the Ridge Regression model will be fitted to the training dataset. The latter consists of the feature variables Xtrain scaled-standardized-so that each of the features will have equal importance concerning its performance in the model. The corresponding target values ytrain join the feature variables in training the dataset:. The model is trained on a dataset to identify such patterns and relationships between the features and target values by minimizing the regularized loss iteratively through optimization.

That is, the model uses input and output data pairs in the training phase and adjusts its internal parameters to best explain or model the training data. This process is repeated until convergence of the model-where further adjustments no longer improve the performance considerably. This will yield a Ridge Regression model that can make predictions on new, previously unseen data by using relationships learned from features to target values.

```
reg = Ridge(alpha=1.0)
reg.fit(X_train, y_train)
```

3.3. EVALUATING MODEL PERFORMANCE:

Basically, two most important metrics when assessing the performance of our trained model will be RMSE and R^2 score. RMSE is defined as the square root of the mean of squared differences between predicted and actual values; it gives an indication of the average magnitude of prediction errors. A lower RMSE means better predictive accuracy. R^2 is defined as a measure of the proportion of variance that is explained by the model, and can be termed as the coefficient of determination. An R^2 score closer to 1.0 means the model explains a high percentage of the variance.

The use of Ridge Regression with the addition of regularization can improve model performance compared to regular Linear Regression. It penalizes large coefficients; therefore, overfitting reduces, yielding more stable and accurate predictions. This makes Ridge Regression yield, in many cases, a lower RMSE and higher R^2 score, indicative of a better predictive reliability and generalization.

```
y_pred = reg.predict(X_test)
RMSE = np.sqrt(mean_squared_error(y_pred, y_test))
R = r2_score(y_pred, y_test)
```

4. RESULTS AND DISCUSSION

Homomorphic encryption knitted into the workflow of machine learning has brought a promising approach toward sensitive computation that can protect data privacy. In this work, we have focused on applying the encryption scheme of Paillier on Ridge Regression while maintaining the privacy of customer data in the process.

4.1. PRESERVATION OF PRIVACY

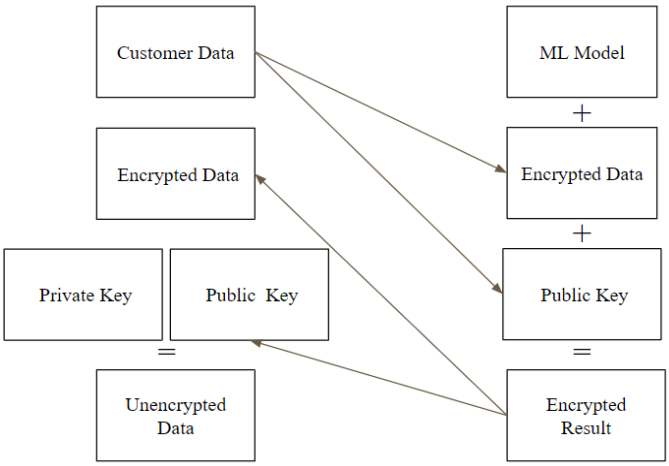
Homomorphic encryption worked well in keeping the data private. We encrypted data before sending for processing, computed on already encrypted data, and avoided any unauthorized access to sensitive information. In this way, it is guaranteed that neither any machine learning service provider nor any other third party can view or access the raw data to maintain strict privacy standards.

4.2. PRACTICAL IMPLICATIONS

Our results further stress the practical feasibility of homomorphic encryption for practical, real-world machine learning applications. Facilities in performing computations over encrypted data without decryption enable an organization to utilize advanced machine learning models effectively while ensuring customer privacy. This result is particularly important in cloud computing environments where security and privacy of data become key considerations.

For example, it may be envisioned in a cloud-based analytics service that customer data is encrypted and sent to the cloud for processing. The cloud service provider performs needed computations over the encrypted data so that raw data remains confidential and inaccessible to the provider. Such a method will facilitate not only the use of sophisticated analytics techniques but also meeting tough data protection regulations and therefore is a very valuable strategy in industries such as finance, healthcare, and personal data management.

5.COMPLETE FLOW DIAGRAM



6. COMPARISON WITH THE EXISTING METHOD

With Homomorphic Encryption:

- Computation Overhead: Operations such as multiplication and summation of encrypted data are usually slower compared with original plaintext operations. This can be due to the additional complexity involved in handling encrypted data. The level of performance degradation usually depends on the encryption scheme chosen along with size of data.
- Memory Usage: Because of the encryption padding and overhead, encrypted data takes up more memory than plaintext data. This increased memory use from encryption can have impacts on overall performance, hence higher usage of resources and delays in processing.

Without Homomorphic Encryption:

- Computational Speed: Operations on the plaintext data are faster, as computing does not have any encryption overheads. The execution of mathematical calculations is directly made over the data, which makes its execution speedier and the processing much more efficient.
- Security-Performance Trade-offs: While the performance of operations on plaintext data benefits from this, the lack of homomorphic encryption thus creates security risks. Indeed, to do processing on sensitive data, it needs to be in plaintext format, thus increasing the possibility of data exposure and possible breaches in computation.

7. CONCLUSION

HE integration into machine learning workflows is a giant leap in maintaining data privacy while enabling secure computations. In particular, our study is using the Paillier encryption scheme in demonstrating that sensitive data protection is guaranteed through all the steps of machine learning, starting from data input to the interpretation of results. Despite the inherent computational overhead and amplified memory consumption entailed by operations on encrypted data, HE serves as a strong approach toward eliminating the security risks of traditional decryption methods. Though still faster to execute, operation execution in plaintext data would be compromising on data confidentiality, thereby exposing it to potential breaches. Thus, HE applications in privacy-preserving machine learning definitely match the strict requirements of data protection and provide a means to apply sophisticated analytical techniques in sensitive environments securely, therefore making this approach an asset to industries reliant on enhanced data security.

8. REFERENCES

[1] SJ Mohammed , DB Taha .“paillier cryptosystem enhancement for homomorphic technique”. Multimed Tools Appl 83, 22567-22579 (2024)

[2] MMS Altaee, M Alanezi, “Enhancing cloud computing security by paillier homomorphic encryption” 2021 - academia.edu.

[3] Munjal, Kundan; Bhatia, Rekha (2022). "A systematic review of homomorphic encryption and its contributions in the healthcare industry".

[4] Sellers, Andrew. "Council Post: Everything You Wanted To Know About Homomorphic Encryption”. *Forbes*. Retrieved 2023-08-18.

[5] P Martins, L Sousa, A Mariano - A survey on fully homomorphic encryption: An engineering perspective. ACM Computing Surveys (CSUR), 2017 - dl.acm.org.

[6] Mohamed Alloghani, Mohammed M. Alani , Dhiya Al-Jumeily ,Thar Baker, Jamila Mustafina,Abir Hussain , Ahmed J. Aljaaf - “A systematic review on the status and progress of homomorphic encryption technologies” 2019 - Elsevier .