Developing Adaptive Homomorphic Encryption through Exploration of Differential Privacy

Yulliwas Ameur*, Samia Bouzefrane
and Soumya Banerjee

CEDRIC Lab, Conservatoire National des Arts et Metiers – CNAM,
Paris, France
E-mail: yulliwas.ameur@lecnam.net
**Corresponding Author*

Received 20 October 2023; Accepted 03 April 2024

Abstract

Machine Learning (ML) classifiers are pivotal in various applied ML domains. The accuracy of these classifiers requires meticulous training, making the exposure of training datasets a critical concern, especially concerning privacy. This study identifies a significant trade-off between accuracy, computational efficiency, and security of the classifiers. Integrating classical Homomorphic Encryption (HE) and Differential Privacy (DP) highlights the challenges in parameter tuning inherent to such hybrid methodologies. These challenges concern the analytical components of the HE algorithm's privacy budget and simultaneously affect the sensitivity to noise in the subjected ML hybrid classifiers.

This paper explores these areas and proposes a hybrid model using a basic client-server architecture to combine HE and DP algorithms. It then examines the sensitivity analysis of the aforementioned trade-off features. Additionally, the paper outlines initial observations after deploying the proposed algorithm, contributing to the ongoing discourse on optimizing the balance between accuracy, computational efficiency, and security in ML classifiers.

Keywords: Machine learning, homomorphic encryption, differential privacy, data security, sensitivity analysis, privacy budget, training dataset, hybrid algorithms, hybrid model.

1 Introduction

Recent scholarly investigations, including those by Li and Micciancio at Eurocrypt 2022 [1], have elucidated that the traditional formulation of security strategies under chosen plaintext attacks falls short in securing approximate homomorphic encryption against diverse adversaries. This research delves deeper into this domain, exploring the extent of attacks and information deviation measurable in a dynamic setup utilizing machine learning methodologies. It is discerned that the information disseminated, retained, and transacted via cloud platforms is susceptible to vulnerabilities owing to the involvement of multiple parties.

To address the vulnerabilities inherent in cloud systems mentioned above, this study explores a range of innovative cryptographic methods, termed Privacy-Preserving Technologies (PPTs). These technologies aim to augment utility by leveraging advanced technologies like cloud computing and machine learning, while maintaining stringent privacy standards. The employed methodology involves post-processing the decryption function output with a mechanism that aligns with a suitable differential privacy (DP) concept, introducing noise proportional to the worst-case error expansion of the homomorphic computation.

In various implementations of privacy-preserving mechanisms, evaluating the distribution of injected and corresponding noises is crucial, even post typical homomorphic encryption noise analysis, due to the potential insignificance of the noise relative to the message. The stability of privacy, analyzed in conjunction with the HE-DP amalgamation and pertinent protocol, is crucial for “Approximate Fully Homomorphic Encryption”. The term ‘approximate’ implies the intentional retention of noise as the least significant bits of the final output during decryption.

The primary objective of integrating HE-DP schemes is to strengthen machine learning applications by providing an additional layer of security against unconventional adversaries. This research establishes a correlation between noise in Homomorphic Encryption and Differential Privacy, exploring this relationship when noise in Homomorphic Encryption is considered as a database-dependent output perturbation. We introduce groundbreaking findings on the guarantees of Differential Privacy through this database addition.

The rest of this paper is organized as follows: Section 2 reviews related works and introduces the mathematical frameworks essential for developing effective machine learning models. Section 3 provides a comprehensive overview of the proposed algorithmic scheme, followed by experimental results in a simulated environment in Section 4. Section 5 presents a model for sensitivity analysis for the mentioned algorithm, and Section 6 concludes with a summary of the contributions, innovations, and potential risks of this scheme, along with potential extensions of these models.

2 Research Motivation

Several recent research endeavors have played a crucial role in aligning the potentials and impacts of homomorphic encryption and differential privacy, serving as the motivation for this study. Xiangyun Tang et al. [1] were pioneers in establishing a well-defined interchangeable property for ML classifiers and ML models in alignment with HE and DP. They introduced Heda, an innovative amalgamation of HE and DP, conceived as a flexible switch to manage the privacy budget and precision parameter tuning to balance the inherent trade-offs Homomorphic encryption is a crucial cryptographic technique that enables computations to be performed directly on encrypted data, thus removing the need for decryption. It serves as a robust solution for preserving the privacy and confidentiality of sensitive information while allowing computations on such encrypted data [5]. This method ensures data security even during processing, reducing dependence on trusted third parties and minimizing the risk of exposing sensitive information to potential threats. As described in [6], a homomorphic map preserves structure, meaning an operation on plaintexts corresponds to an operation on ciphertexts. This implies that altering the sequence of operations preserves the outcome post-decryption, i.e., ‘encrypt-then-compute’ and ‘compute-then-encrypt’ yield equivalent results.

Aligned with the trend of parametric models, Bossuat et al. (2022) [3] optimized their model of approximate homomorphic encryption, reducing the occurrences of failures and enhancing precision through distributed and random encapsulation, initiated with bootstrapping. Preliminary investigations

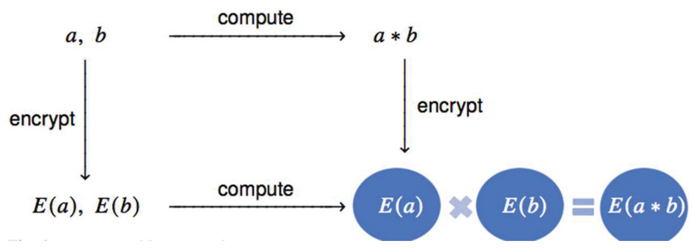


Figure 1 Homomorphic map [6].

have uncovered additional hybrid cryptographic techniques, such as Lattigo [4], which integrates intriguing multiparty primitives with the HE protocol in lattice-based cryptography.

Motivated by these advancements, this paper pursues the design and validation of a refined HE-DP model, incorporating sensitivity analysis to ensure balanced noise insertion to mitigate various trade-offs. The motivation is driven by the desire to explore and amplify the synergistic capabilities of homomorphic encryption and differential privacy, contributing to the progressing field of cryptographic research.

2.1 Classical Homomorphic Encryption

Homomorphic encryption is a crucial cryptographic technique that enables computations to be performed directly on encrypted data, thus removing the need for decryption. It serves as a robust solution for preserving the privacy and confidentiality of sensitive information while allowing computations on such encrypted data [5]. This method ensures data security even during processing, reducing dependence on trusted third parties and minimizing the risk of exposing sensitive information to potential threats. As described in [6], a homomorphic map preserves structure, meaning an operation on plaintexts corresponds to an operation on ciphertexts. This implies that altering the sequence of operations preserves the outcome post-decryption, i.e., ‘encrypt-then-compute’ and ‘compute-then-encrypt’ yield equivalent results.

There are different types of homomorphic encryption schemes, each with its own properties and capabilities [7]:

- **Partially Homomorphic Encryption [8]:** This type of homomorphic encryption allows computation on either addition or multiplication operations, but not both simultaneously. For example, the Paillier encryption scheme is partially homomorphic under multiplication, while the El Gamal encryption scheme is partially homomorphic under addition.

- **Somewhat Homomorphic Encryption (SHE):** Somewhat homomorphic encryption schemes support a limited number of both addition and multiplication operations. However, the computation capabilities are constrained even if they still provide practical utility. Examples of SHE include the Gentry-Halevi encryption scheme and the BFV encryption scheme [9].
- **Fully Homomorphic Encryption (FHE):** Fully homomorphic encryption schemes allow for arbitrary computations on encrypted data, including both addition and multiplication operations. FHE is the most powerful form of homomorphic encryption but comes with more complexity and computational overhead. Example of FHE is BGV encryption scheme [10].

2.2 Relevant Mathematical Perspectives

The proposed model envisages the combination of a given approximate FHE scheme with the tool of differential privacy. The construction can be given as follows:

- Given an approximate FHE scheme, we modify the decryption function by post processing its output (similar to decrypted messages) with an appropriate chosen Differential privacy model.

Here, we can consider two levels:

- Approximate FHE with a static noise: this instance describes that bound can be truly computed as a function of homomorphic encryption. Finally, this could be used on the input cipher text.
- Approximate FHE with a dynamic noise. In this case, the bound can be computed by the decryption function. In turn, the decryption function allows the input, the cipher text and secret key.

We deploy the proposed model HDP (Homomorphic Differential Privacy), where we analyze a DP model while adding Gaussian dynamic noise to the input. Hence, let $\Pi = (KeyGen, Ence, Dec, Eval)$ be an FHE scheme with a plain text space:

$\dot{S} \subsetneq s$, where $\widetilde{S} \subseteq \tilde{Z}$ is normal space with normal $||\cdot||$ operator.

We also consider a policy to add Gaussian noise with this text space and normal $||\cdot||$ operator. We also assume a standard deviation norm to add Gaussian noise at later stage to test the level of privacy. Therefore, for any deviation $\partial > 0, n \in N$, the *Kullback-Leibler Differential Privacy* [16] could be a dynamic method to add the noise. However, for the presented model we follow standard gaussian noise insertion mechanism.

2.3 Mathematical Assumptions and Existing Libraries

2.3.1 Gaussian noise: maintaining privacy and preserve statistical properties

Gaussian noise can be used as a mechanism to add differential privacy to a dataset. Gaussian noise is a type of random noise that follows a normal distribution, and it can be added to numerical data in order to mask the original values while still preserving the statistical properties of the data.

In the context of differential privacy, Gaussian noise can be added to the data in order to make it more difficult to identify individual records or extract sensitive information from the dataset. The amount of noise added can be controlled by adjusting the standard deviation of the Gaussian distribution with higher standard deviations resulting in more noise and greater privacy protection. However, adding too much noise can make the data less useful for analysis or modelling. So, it is important to strike a balance between privacy and utility when using this technique. For the use case, a standard deviation of 0.1 is used for generating noises. This value can be increased further but will result in a significant decrease in terms of accuracy.

2.3.2 Suitable & most appropriate encryption libraries

All HE schemes have common steps: key generation, encryption, decryption and homomorphic operations on the ciphertexts.

Mathematically, the security of Paillier assumes to be synchronized with the hardness of factoring.[1] Referring a pair of ciphertext (c_1; c_2) is (m_1; m_2) under the same Paillier encryption, with M as public key, then

$$c_1 \times c_2 = [\![(1+M)^{m1+m2} r^N mod M^2]\!], \text{ where } (m1 + m2) < N.$$

In the context of this project, the *Paillier encryption scheme* is chosen as the preferred homomorphic encryption technique for several elaborated reasons as explained in the next section.

2.4 *Paillier Cryptosystem*: Scheme and Properties

The Paillier cryptosystem is a public key cryptosystem used for encryption and decryption of data. The cryptosystem is **asymmetric**, meaning that it uses two keys: a **public key** and a **private key**. The public key is used for encryption while the private key is used for decryption. The Paillier

[1]Definition: Any computational adversary given as input N, the product of two random n-bit prime numbers, shall not be able to factor it.

Table 1 Summarizes the most implemented and studied schemes by the cryptographic community

Library	ZAMA[1]	SEAL [17]	Paillier [8]
Encryption	Homomorphic encryption with limited operations	Fully Homomorphic Encryption (FHE) capabilities	Partial Homomorphic Encryption with limited support
Encryption Mechanism	Efficient performance with optimized resource usage	High-performance encryption operations	Moderate performance due to limited homomorphic operations
Geolocation	Limited support for geolocation tasks	No specific support for geolocation	Suitable for geolocation tasks, enabling homomorphic geofencing
Geo-fencing	Limited support for geofencing algorithms	No specific support for geofencing algorithms.	Possible to implement geofencing algorithms with custom methods.
General Limitations	Limited support for complex homomorphic operations	Requires more computational resources.	Limited support for complex homomorphic operations.

https://www.zama.ai/.

cryptosystem has several unique properties, including its ability to perform **homomorphic addition** and **scalar multiplication**, which means that it can perform addition and multiplication operations on ciphertexts without the need to decrypt them as first step. This property makes it useful for privacy-preserving computations.

$$D(E(A) + E(B)) = A + B$$

Or

$$D(E(A) \times Scaler) = A \times Scaler$$

In the above depiction, **A** and **B** and **Scalar** are actual numbers. **E (...)** is the *Paillier encryption function*. **D (...)** is the Decryption operation. The system is **semantically** secure, which means that an attacker cannot learn any information about the plaintext from the ciphertext. For the study, the *Paillier Cryptosystem* is leveraged by HE-DP algorithm to ensure secured data transfer and computations with privacy. In the following section, two separate high level descriptions of HE-DP (e.g. client & server architecture) are presented (See Figure 2).

Client-Side Algorithms:

```
Process:
1: Generate Paillier public-private key pair using the Paillier Cryptographic Scheme
2: Save the public-private key as a JSON
3: Create a PaillierPublicKey object pub_key with the value of 'n' from the public key
4: Create a PaillierPrivateKey object priv_key with pub_key, p, and q from the private key.
5: For each data in the dataset, do
6: Add Gaussian noise at randomized intervals
7: Encrypt the data with pub_key
8: End for
9: Convert the encrypted data to JSON
10: Append pub_key with JSON
11: Send the JSON to the SERVER side
12: While no JSON prediction from the server, do
13: Wait
14: End while
15: If pub_key of the response matches the generated pub_key
16: For data in JSON prediction, do
17: Decrypt the data with priv_key
18: End for
19: End if
```

This algorithm outlines the steps involved in securing and transmitting data from the client's side using the Paillier Cryptographic Scheme.

Server-Side Algorithms

```
Process:
1: While there is no JSON request from the client, wait
2: End while
3: Load trained weights from the base model
4: For each data in the JSON request, do
5: If the data is equal to pub_key, then
6: Create a PaillierPublicKey object pubkey with the value of 'n' from pub_key
7: Else
8: Convert data to Paillier EncryptedNumber objects
9: Calculate the prediction by taking the dot product of loaded weights and
EncryptedNumber objects
10: Assign the result to prediction and append pub_key (JSON)
11: End if
12: End for
13: Send the result JSON to the CLIENT side
```

This algorithm demonstrates the server-side operations upon receiving encrypted data from the client, utilizing the Paillier Cryptographic Scheme for secure data processing.

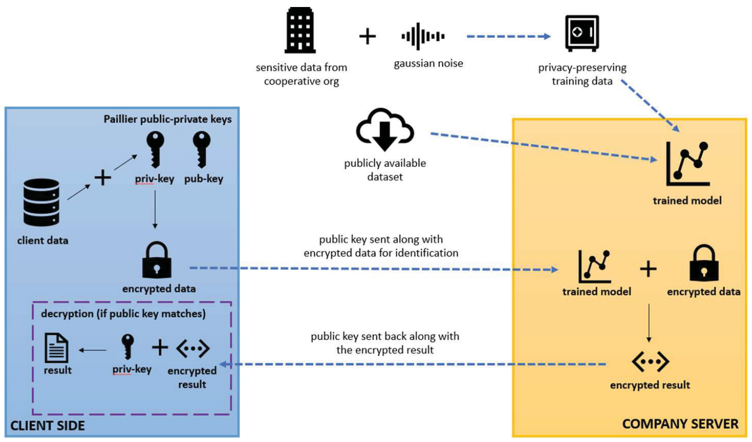

sensitive data from cooperative org
gaussian noise
privacy-preserving training data
Paillier public-private keys
priv-key
pub-key
client data
encrypted data
publicly available dataset
trained model
public key sent along with encrypted data for identification
trained model
encrypted data
decryption (if public key matches)
result
priv-key
encrypted result
public key sent back along with the encrypted result
encrypted result
CLIENT SIDE
COMPANY SERVER

Figure 2 Client-server HEDP interaction schema.

2.5 Experimental Validation and discussion on Results

Combining homomorphic encryption and differential privacy provides an even more powerful tool for secure machine learning. By combining these two techniques, it is possible to perform computations on encrypted data while also ensuring that the privacy of the individuals in the data set is protected.

In this paper, a novel measurement is proposed and an evaluation of the performance of Homomorphic Encryption with Differential Privacy (HEDP) has been done on a *Breast Cancer Wisconsin* dataset.

2.5.1 Proposed *HEDP*: architecture, process and code walkthrough

Based on the dataset, the objective is to predict whether the type of cancer is *Malignant* or *Benign* based on the attributes provided in the dataset. It is to be kept in mind that the flow of machine learning model training and prediction should proceed in such a manner that it preserves both privacy and security. To ensure the maximum security and privacy, HEDP algorithm is proposed. The methodology starts by training a base model on a publicly available dataset. This dataset can be used by any organization or individual who wishes to train a ML model. However, if the dataset is sourced from a sensitive organization, then privacy-preserving measures need to be taken. This is where differential privacy comes in.

Gaussian noise is added to the sensitive data to preserve privacy. The amount of noise added depends on a privacy parameter, which is set by the organization or individual providing the sensitive data. Once the privacy-preserving modifications are made to the dataset, the base model is trained on this modified dataset. This base model is used by any client who wants to make predictions on their own sensitive data. However, the client does not wish to reveal their sensitive data to the party (company offering ML services). To further enhance the privacy, Homomorphic Encryption is added into the mix, particularly the Paillier encryption scheme. The client generates a Paillier public-private key pair and uses the public key to encrypt their data. The public key is then attached along with the data and sent to the ML company over a secure network.

Once the ML component receives the encrypted data, it uses the public key to extract Paillier encrypted objects. Then it performs machine learning computations on the Paillier objects using the already trained base model to predict the output. The output is an encrypted prediction. This is because the encrypted data is encrypted with Paillier cryptosystem, which is a homomorphic encryption scheme. Any sort of addition and scalar multiplication done on this encrypted data is the same as doing the same operations on the decrypted data.

The encrypted prediction is sent back to the client. The client then decrypts it with their Paillier private key and gets their result.

2.6 Scopes of Implementations

In this experiment, the dataset used is *Breast Cancer Wisconsin*. The Breast Cancer Wisconsin (Diagnostic) dataset is a widely utilized dataset in machine learning and data mining research. It was initially created by Dr. William H. Wolberg, a physician at the University of Wisconsin Hospital at Madison, Wisconsin, USA. The data was gathered by examining fine needle aspirates (FNA) of breast masses, each of which was labeled as either malignant or benign.

This dataset encompasses features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These features describe various characteristics of the cell nuclei, such as radius, texture, perimeter, area, smoothness, compactness, concavity, symmetry, and fractal dimension. These attributes aid in characterizing the cell nuclei as either benign or malignant. The dataset is publicly available, for the demonstration purposes, it is assumed that the data received is pre-processed by injecting Gaussian noise.

To increase the strength of the Gaussian noise added to the dataset, it is required to increase the value of the standard deviation (sigma) parameter of the Gaussian distribution used to generate the noise. This parameter controls the spread of the distribution, so higher values of sigma will generate noise with greater magnitude. Different values of sigma can be experimented to find the optimal level of noise for the specific use case. However, it is crucial that that adding too much noise may adversely affect the accuracy of the given machine learning model. The following sub-section describes the deployment process for both client and server architecture, which is essential for experimental validation.

2.6.1 Client-Side Code Walkthrough

The following lists of functions describe the client-side deployment:

- ***storeKeys( )*:** This function generates public and private keys based on the Paillier Homomorphic Encryption Scheme, and saves the keys in a JSON file named *client_public_private_keys.json*.
- Content of *client_public_private_keys.json*
- ***getKeys( )***: This function reads the *client_public_private_keys.json* file and creates public and private key objects using the Paillier module.
- ***serializeData( )*:** This function takes in the *public_key* object and *data* as inputs, encrypts the data using the public key, and returns a serialized JSON object. The encrypted data is stored as a list of tuples with the ciphertext and exponent.

The encrypted data which is sent to the server looks like the following:

- ***load_prediction()*:** This function reads a JSON file named *prediction.json* and returns its contents as a dictionary.
- Data contents of *prediction.json*

The *load_prediction()* function then loads a pre-generated encrypted prediction from the server and decrypts it with the private key of Paillier scheme. The resultant answer is a Regressed Result (more about why is it not a binary result in the server-side code walkthrough: *computeData()* function). The regressed result is passed through a Sigmoid function which squeezes the regressed result strictly between 0 and 1. The scheduled schema is :

0 □ Benign Cancer

1 □ Malignant Cancer

2.6.2 Server-side code walkthrough

Firstly, a base model is trained on the privacy-preserved data (in our case). There will not be much difference in training the base model. The process is the same. Once the model is trained, the trained weights are saved and used for homomorphic computations.

The *trained_weights* contain the weights fitted by the model on the breast cancer dataset. The following lists of responsible functions are mandatory:

- ***getData():*** retrieves the encrypted data from the client and loads it into the system.
- ***computeData( )*:** This function utilizes *trained_weights* to perform a dot product with encrypted Paillier objects. This is a homomorphic scalar multiplication, which is a valid operation on the Paillier encryption scheme. However, the result is not in binary format and requires conversion to binary using the sigmoid function. It is important to note that Paillier Encryption Scheme is a Partially Homomorphic Scheme due to the fact that it cannot handle division homomorphism.

Even mimicking the division with multiplicative inverse results in inaccurate encrypted notation. Therefore, to obtain accurate results, the sigmoid function is employed on the client side. The serialization of output function will follow:

- ***computeAndSerializeData( )*:** formats the encrypted results along with the respective public key.
- ***save_prediction( ):*** saves the prediction in *prediction.json* file. This is the predicted result sent to the client for decryption.

2.7 Performance Analysis: Proposed HEDP versus Standard Algorithms

The analysis comprises of the plot for *Accuracy vs Iterations*.

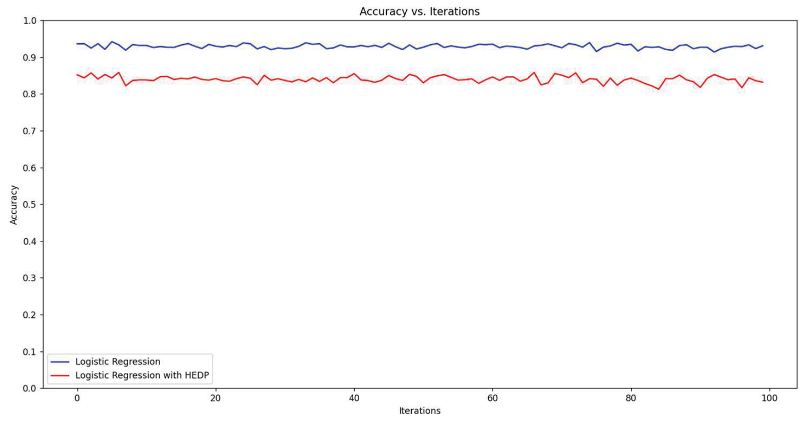


Figure 3 Accuracy versus iteration plot.

In Figure 3, it is demonstrated that the observed loss is not significant since the training and prediction methods did not use any lossy encryption schemes. The accuracy decline in HEDP can be attributed to the noisy dataset used for training. However, it is noteworthy that the model performed remarkably well, considering the fact that it was trained entirely using privacy-preserving techniques. During the execution phase, the CPU consumptions from both the sides (client and server) were observed.

2.7.1 The client-side plot

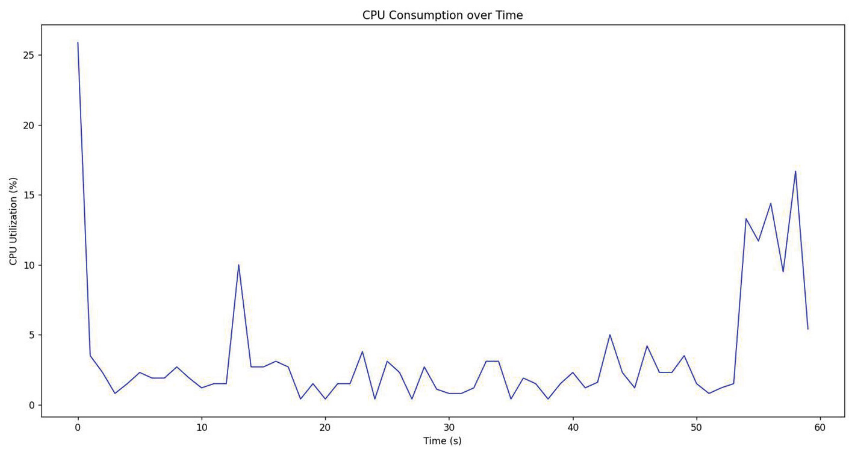


Figure 4 CPU utilization with time occupancy (client side).

Observations:

– The initial spike was due to Paillier encryption, where the client generated Paillier public, private keys and then used it to encrypt the data it intended to send to server.
– The sudden spike at the end is attributed to decryption (using private key to decrypt the result) as well as the sigmoid operation on the received result. Here, the maximum CPU utilization observed is 25%.

2.7.2 Server-side plot

Observations

– The server is idle initially. In Figure 5, as soon as it received the data, the computational process kicked in (graph peaks consistently after 35s mark).

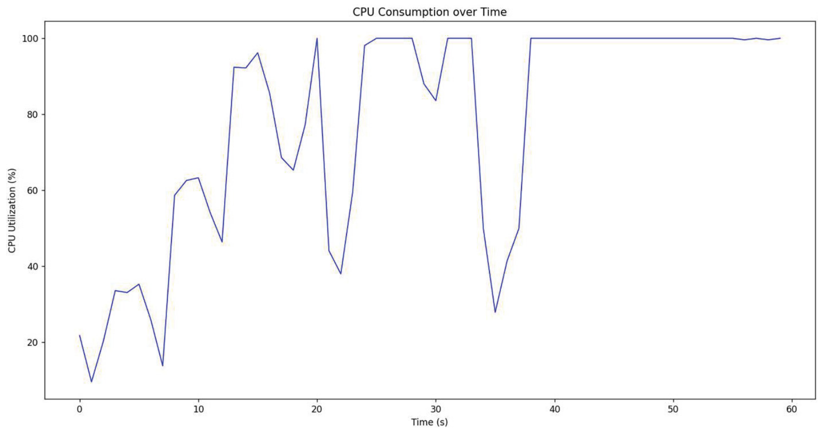


Figure 5 CPU utilization with time occupancy (server side).

- The computational process is heavy as it is evident in the plot. It rapidly consumed around 100% of CPU utilization for homomorphic operations on encrypted data.
- Thus, maximum CPU Utilization reaches to 100%.

2.8 Standard Algorithm (Linear Regression without HEDP) CPU Plot

The following plot demonstrates the CPU utilization of the standard algorithm [15], which was trained on the same server.

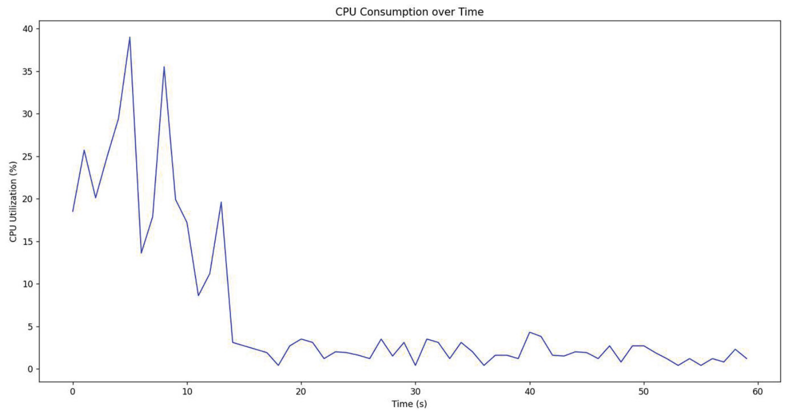


Figure 6 Standard CPU utilization without HEDP scenario.

Observations

– In Figure 6, the training process lasted only a few seconds (7s).
– The peak CPU utilization is around 40%. This is much lesser than the HEDP counterpart.
– The CPU utilization drops quickly after the training process is complete, as there is no data transmission over the network involved.

2.9 Sensitivity Analysis

Motivated by the phenomenal work by Tabitha Ogilvie [21], where the theme has been throughly investigated that dependence of HE noise on the underlying data as a critical barrier to privacy, and derive new results on the Differential Privacy under the constraints of native noise impact on HE.

This work is pivoted on the central facts e.g. sensitivity, noise variance and message dependence and therefore the paper pointed out the extent to which the noise growth in homomorphic encryption can provide differential privacy to the output. Finally, the paper solicits properly account for message dependence, the privacy leakage is much higher, and also it has been found that message dependence dominates noise growth. The base of the paper inspired the sensitivity analysis derivation in more simplistic way for this present work.

Sensitivity analysis [12] is expressed as the relation to the Laplace mechanism privacy budget and the justified sensitivity to regulate the amount of noise addition. Therefore, every HEDP algorithm should deduce suitable mathematical expression to fix privacy budget formula and accommodate most likelihood sensitivity procedure for DP mechanism.

Without the loss of generality, we assume that to guarantee the efficacy of Differential Privacy if considered homomorphically, then it is expected to consider only the privacy measure of terminal output.

In few cases, the entropy of terminal output follows:

$$[\alpha + N(0, \partial^2)]. \tag{1}$$

where α is the "true" output of the algorithm. Thus, an initial approach may attempt to ensure ρ is large enough to mask the difference $(\alpha - \alpha')$ over adjacent databases.

In this work, we consider failure probability δ of DP under the real variance of HE model, where sensitivity δ_f could vary with the value of variance in proportion with a monotonically increasing function. However, the variance of an algorithm's output could be more prominent numerically when it is evaluated homomorphically (this behavior is dependent on the input data). Therefore, the proposed algorithm presented here is to model the output from the database with proper noise distribution. Thus, this sensitivity analysis comprises of database and noise analysis.

The following core theorem can be described as:

Let ε (privacy budget for the proposed HE-DP algorithm including client and server side deployment) $\in$ (0, 1) be arbitrary. For $c_2 > 2$ ln (1.25/δ), the Gaussian Mechanism is (ε, δ)-differentially private whenever $\partial \geq$ cd$_f$ /ε, where δ_f is the sensitivity.

We consider two databases PD *(Pilot database*) and RD *(Reference database)* and let A be the algorithm output when we use PD as input database, and A' when we use *Reference database* RD$'$.

Our strategy will be to show the ratio of probability density functions as:

$$\frac{F_{PD}}{F_{RD}}\, A(\alpha)/fA'(\alpha) > e^{\varepsilon}, \tag{2}$$

except with probability at most δ as α follows the distribution of the proposed HEDP algorithm A.

2.10 Exceptions in the Proposed HEDP Model

In the proposed model with HE, the multiplication will be automatically be the part of plain texts and thus it will impress on the variance of $(\alpha - \alpha')$. Therefore, we assume polynomial model as $P\sigma_1^2\sigma_2^2$ and thus added entity with plaintexts t$_1$ and t$_2$ can be expressed as:

$$P\sigma_1^2\sigma_2^2 + \sigma_1^2\|t_1\|^2 + \sigma_2^2\|t_2\|^2, \tag{3}$$

Therefore, for the given homomorphic encryption, input database should be influential with respect to the given entropy followed:

$$[\alpha + N(0, \partial^2)]. \tag{4}$$

As we consider $(\alpha - \alpha')$ over adjacent databases, therefore to tune with the consulting databases focused for training, the objective is to fix the Gaussian mechanism for the proposed HEDP algorithm in such a way that the variance ∂ will depend on the combined value of (ε, δ) and it will be evaluated as the following:

$$2ln\left(\sqrt{\frac{prob1}{prob2}}\right) > 1 \tag{5}$$

the value of ε also varies with noise scale. If we consider κ as coefficient which will support to express the agreement of the binary outcome of the two training databases, where the noise ε is added to achieve the differential privacy for different mathematical operations relevant to homomorphic encryption.

However, to find out the ratio of the probability of given noise function, we follow the following formula:

$$\frac{F_{PD}}{F_{RD}} = exp\left(\frac{\|\gamma - \alpha'\|^2}{2\partial^2} - \frac{\|\gamma - \alpha\|^2}{2\partial^2}\right) \tag{6}$$

$$= exp\left(\frac{1}{2\partial^2} + (\|\gamma - \alpha + \kappa\|^2 - \|\gamma - \alpha\|^2)\right) \tag{7}$$

$$= exp\left(\frac{1}{2\partial^2} + 2(\gamma - \alpha) \cdot \kappa + \|\kappa\|^2)\right) \tag{8}$$

It signifies that the database for training the database with deliberate noise needs a *Pilot database (PD)*, where the proposed algorithm A and its variance A' will be functional to another parallel refernce database with a relational expresssion as: $A \sim N(\alpha, \Sigma)$, here Σ can find its contemporary diagonal element Σ'.

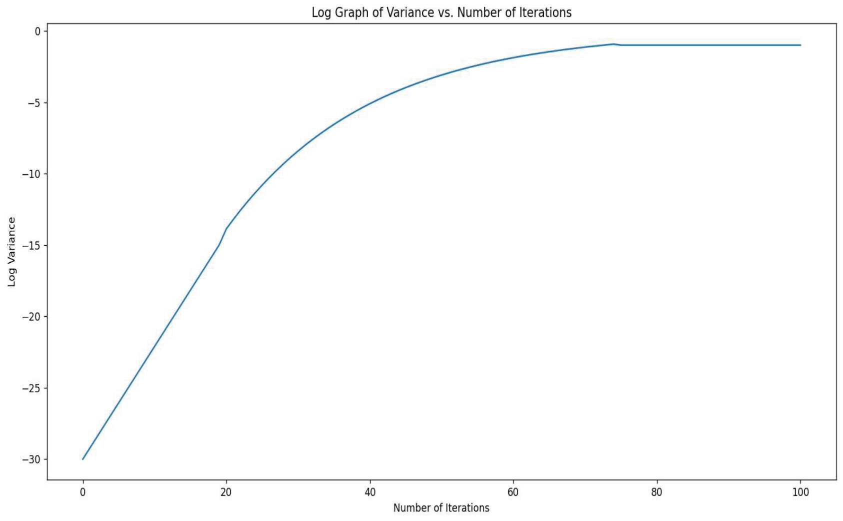
Thus,

$$\frac{F_{PD}}{F_{RD}} = \prod_{i=1}^{PD} \frac{\partial RD}{\partial PD} exp \frac{1}{2}\left(\frac{\gamma_i - \alpha_i'}{\partial RD}\right)^2 - \left(\frac{\gamma_i - \alpha_i}{\partial PD}\right)^2\Bigg) \tag{9}$$

Hence, with log sensitivity and iterations for the variance, we will be interested to evaluate maximum likelihood probability and we can rewrite the expression as:

$$\frac{F_{PD}}{F_{RD}} = \prod_{i=1}^{PD} \frac{1}{\zeta_i} exp\left(\frac{1}{2}\sum_{i=1}^{PD} \zeta^2 \left(\frac{\frac{(\gamma_{i-}\alpha_i)}{\partial_i} - \kappa_i}{}\right)^2 - \left(\frac{\frac{(\gamma_{i-}\alpha_i)}{\partial_i}}{}\right)^2\right. \tag{10}$$

This is in tune with polynomial model of $P\sigma_1^2\sigma_2^2$ for HE multiplication. It also highlights that in the present simplistic model rescale and squaring have not been addressed. After inclusion of these components of noise growth the maximum likelihood probability of log variance could be changed accordingly.



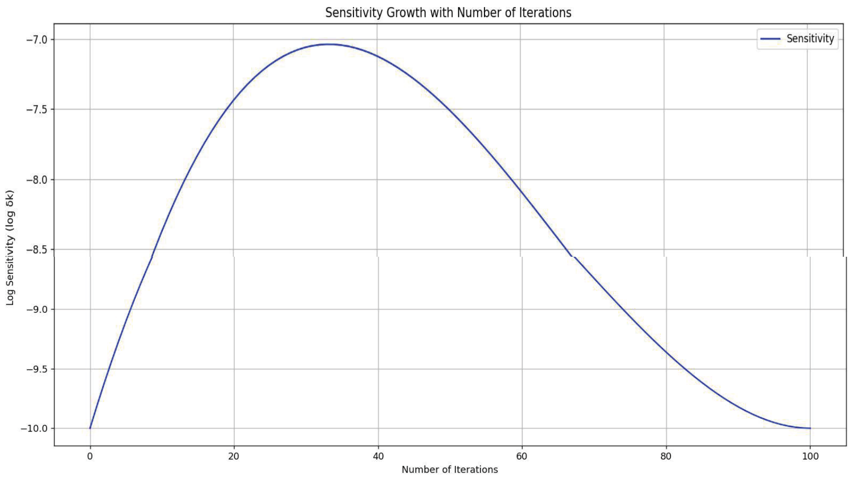


Figure 7(a) & 7(b) Sensitivity observation of proposed HEDP.

Here, ζ_i is the relation between diagonal entries for the pilot and referential database, subjected for the training and sensitivity κ should be in the appropriate range of variance of $(\alpha - \alpha')$.

Following the formulated Equations (8), (9) and (10) above, the validation of sensitivity analysis for proposed HEDP algorithm can be done. We observe that there are two phases to measure the trend of the system: (a) log variance with iterations. (b) log sensitivity with iterations.

Significantly, in the first plot [Figure 7(a)], it is observed that log variance inclines towards positive training value after certain number of iterations. Thus, training accuracy with noise becomes more consistent with positive value as it grows up with number of iterations. However, proposed HEDP in Figure 7(b) demonstrates to attain a peak value with log sensitivity scale -7 in 40th iterations. The curve flattens more as it grows up with a greater number of iterations. It signifies that trade-off between accuracy and computational speed up in the proposed model becomes stable with the present use-case. However, to find out more robust sensitivity, the value of Gaussian noise, the noise variance and dependency analysis have also to be performed on the HEDP algorithm.

2.11 Conclusion and Scope of Further Research

In this paper, a rudimentary algorithm for combining homomorphic encryption and differential privacy is proposed to demonstrate an effective strategy towards the precession, computational efficiency and privacy budget trade-off. It is emphasized with the deployment of such a proof-of-concept demonstrated with an elementary client-server architecture, which could position suitable justification of library compatibility for HE environment. We observe that, in DP, the main challenge is to reduce the tradeoff between privacy and accuracy. To address this challenge, future research can explore the development of more robust mechanisms that add less noise while providing more privacy guarantees. This is indicative for complete sensitivity analysis of such HEDP protocol to perform noise analysis and to check message dependency.

Additionally, it is possible to propose different relaxations for DP in the context of distributed paradigm or to boost DP using other techniques like anonymization, subsampling or cryptography. For HE, the major component is to reduce the tradeoff between privacy and computational complexity. To address this challenge, the research should focus on accelerating HE primitives while identifying algorithmic approaches to reduce the complexity of certain operations, such as division. By improving the efficiency of HE, privacy can be maintained while mitigating the computational overhead. Furthermore, the combination of HE and DP is also an interesting direction. However, combining these techniques is not trivial, as it requires reducing the tradeoff between computational complexity, model accuracy, and privacy. As suggested in the work of Sébert et al. [14], combining these two techniques may offer a way to protect the raw data from all the participants in the distributed process.

There are few research initiatives where a private routing protocol that can be used to communicate anonymously between different networks. For example, the protocol mentioned in the paper can be applied in a variety of Internet of Things scenarios: from Wireless Sensor Networks, to interconnected IoT systems composed of different devices or infrastructures. Finally, this protocol achieves context privacy by using homomorphic encryption, tunneling, and the spatial Bloom filters [18].

In the context of more searchable encryption, it is evident that online social networks and IoT applications can be orchestrated with specialized encryption techniques. The approach facilitates a more controlled means for data dissemination in a public OSN. Predicting trust in an online social network is a challenging task due to the lack of physical connectivity and complete factual information. A selective inconspicuousness method is proposed. It first identifies the personally identifiable information from tweets and user bio, anonymizes it, and transfers the crucial data via the cloud, from where authorized users can retrieve them via the proposed searchable encryption mentioned in the paper [19].

Furthermore, considering the advancements made by Ameur et al. (2022) in developing a secure and non-interactive k-NN classifier using symmetric fully homomorphic encryption, it would be intriguing to explore the adaptation of a hybrid version of this classifier using homomorphic encryption and differential privacy. Such an adaptation could optimize the classifier's performance and facilitate a scalable deployment of k-NN on encrypted data, thereby enhancing data security in statistical database systems.

References

[1] Li, B., Micciancio, D., Schultz, M., Sorrell, J.: Securing approximate homomorphic encryption using differential privacy. In: Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara CA, USA, August 15–18, 2022, Proceedings, Part I. pp. 560–589. Springer (2022).

[2] Tang, Xiangyun, et al. "When homomorphic cryptosystem meets differential privacy: training machine learning classifier with privacy protection." *arXiv preprint arXiv:1812.02292* (2018).

[3] Bossuat, J.P., Troncoso-Pastoriza, J., Hubaux, J.P.: Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation. In: Applied Cryptography and Network Security: 20th International Conference, ACNS 2022,

Rome, Italy, June 20–23, 2022, Proceedings. pp. 521–541. Springer (2022).

[4] Lattigo v2.2.0. Online: http://github.com/ldsec/lattigo (Feb 2023), ePFL-LDS.

[5] Kiesel, R.; Lakatsch, M.; Mann, A.; Lossie, K.; Sohnius, F.; Schmitt, R.H. Potential of Homomorphic Encryption for Cloud Computing Use Cases in Manufacturing. *J. Cybersecur. Priv.* **2023**, *3*, 44–60. https://doi.org/10.3390/jcp3010004.

[6] Kristin E. Lauter, Private AI: Machine Learning on Encrypted Data, International Association for Cryptologic Research, https://eprint.iacr.org/2021/324.pdf.

[7] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti, 2018. A survey on homomorphic encryption schemes: theory and implementation. ACM Comput. Surv. 1, 1, Article 1, January **2018**, pp. 35.

[8] Nassar et.al, Paillier's encryption: Implementation and cloud applications, 2015 International Conference on Applied Research in Computer Science and Engineering (ICAR).

[9] Wibawa, F.; Catak, F.O.; Sarp, S.; Kuzlu, M. BFV-Based Homomorphic Encryption for Privacy-Preserving CNN Models. *Cryptography* **2022**, *6*, 34. https://doi.org/10.3390/cryptography6030034.

[10] K. Hariss, M. Chamoun and A. E. Samhat, "On DGHV and BGV fully homomorphic encryption schemes," *2017 1st Cyber Security in Networking Conference (CSNet)*, Rio de Janeiro, Brazil, **2017**, pp. 1–9, doi: 10.1109/CSNET.2017.8242007.

[11] Kim, S., Park, M., Kim, J., Kim, T., Min, C.: Evalround algorithm in ckks bootstrapping. In: Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security Taipei, Taiwan, December 5–9, 2022, Proceedings, Part II. pp. 161–187. Springer (2023).

[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in Proceedings of the Third Conference on Theory of Cryptography, ser. TCC'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 265–284.

[13] Costache, A., Nu¨rnberger, L., Player, R.: Optimizations and trade-offs for helib. Cryptology ePrint Archive (2023).

[14] Sébert, A.G.; Sirdey, R.; Stan, O.; Gouy-Pailler, C. Protecting Data from all Parties: Combining FHE and DP in Federated Learning 2022. arXiv:2205.04330 [cs].

[15] Kifer, D., Smith, A., Thakurta, A.: Private convex empirical risk minimization and high-dimensional regression. In: Mannor, S., Srebro, N., Williamson, R.C. (eds.) Proceedings of the 25th Annual Conference on Learning Theory. Proceedings of Machine Learning Research, vol. 23, pp. 25.1–25.40. PMLR, Edinburgh, Scotland 25–27th Jun 2012.

[16] Securing Approximate Homomorphic Encryption Using Differential Privacy. ePrintBaiyu Li, Daniele Micciancio, Mark Schultz, Jessica SorrellCRYPTO 2022.

[17] Peng, Zhiniang. "Danger of using fully homomorphic encryption: A look at Microsoft SEAL." *arXiv preprint arXiv:1906.07127*, **2019**.

[18] Palmieri P, Calderoni L, Maio D. An Anonymous Inter-Network Routing Protocol for the Internet of Things. JCSANDM [Internet]. 2017 Apr. 16 [cited 2023 Nov. 4];6(2):127–146. https://journals.riverpublishers.com/index.php/JCSANDM/article/view/5213.

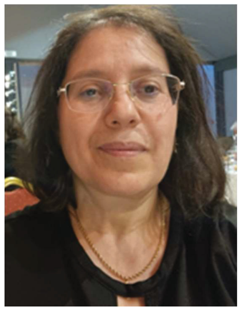
[19] Shetty NP, Muniyal B, Yagnik N, Banerjee T, Singh A. A Privacy Preserving Framework to Protect Sensitive Data in Online Social Networks. JCSANDM [Internet]. 2022 Nov. 7 [cited 2023 Nov. 4]; 11(04):575–600. https://journals.riverpublishers.com/index.php/JCSANDM/article/view/12461.

[20] Ameur, Y., Aziz, R., Audigier, V., Bouzefrane, S. (2022). Secure and Non-interactive k-NN Classifier Using Symmetric Fully Homomorphic Encryption. In: Domingo-Ferrer, J., Laurent, M. (eds) Privacy in Statistical Databases. PSD 2022. Lecture Notes in Computer Science, vol. 13463. Springer, Cham. https://doi.org/10.1007/978-3-031-13945-1_11.

[21] Ogilvie, T. (2023). Differential Privacy for Free? Harnessing the Noise in Approximate Homomorphic Encryption. Cryptology ePrint Archive, Paper 2023/701. Retrieved from: https://eprint.iacr.org/2023/701.

Biographies

Yulliwas Ameur received his Master's degree in "Mathematics of Cryptography and Communications" from the University of Paris 8 in 2019. He is a member of ARCSI, a leading French association in the fields of cryptography and digital security. During his master's internship at Inria Rennes – Bretagne Atlantique, he worked on resilience against covert channel attacks on code-based cryptographic schemes. He recently completed his Ph.D. at CNAM, under the supervision of Samia Bouzefrane and Vincent Audigier, focusing on homomorphic encryption and machine learning. Yulliwas is an expert in cybersecurity and serves as a trainer at several universities and engineering schools across Europe.



Samia Bouzefrane received the Ph.D. degree in computer science from the University of Poitiers, France, in 1998. After four years at the University of Le Havre, France, she joined the CEDRIC Lab of Conservatoire National des Arts et Métiers (Cnam), Paris, in 2002. She is currently full professor in Cnam. She is the coauthor of many books (Operating Systems, Smart Cards, and Identity Management Systems). She has coauthored more than

120 technical articles. Her current research interests include the Internet of Things and security using AI techniques.

Soumya Banerjee, SM-IEEE was an invited Research professor and at present as Senior Associated Researcher at INRIA–AIO (the French National Institute for Computer Science https://aio.inria.fr/team/) Paris since November 2018 Conservatoire National des Arts et Métiers (CNAM), Laboratoire CEDRIC.

From 2019 (https://cedric.cnam.fr/lab/author/banerjs/), he is also acting as the Chief Technology Officer of *Mext-Metaverse, Paris,* France (https://mext.app/) along with associated senior researcher activities & projects at INRIA Paris(mainly on various use-cases for different variations Of Reinforcement Learning. At *Mext*, he is developing of Deep hybrid learning based recommendation and unsupervised machine learning for business Eco-system and communication systems. In addition to, he is the Senior Vice President & Research & Innovation of Trasna Solutions Ltd (Europe https://www.trasna.io/) for embedded intelligence with ML accelerator for RISC V and Private Blockchain implementations on chip. From 2021–2022, he was involved with the center of excellence with UCC Cork and Govt. of Ireland. He is having several projects and product implementations on private Blockchain, e-SIM and smart manufacturing & logistics, NFT and data analytics.