Received: 13 July 2023 | Revised: 25 December 2023 | Accepted: 10 January 2024

DOI: 10.1002/spy2.372

RESEARCH ARTICLE

WILEY

# Privacy-preserving credit risk analysis based on homomorphic encryption aware logistic regression in the cloud

**V. V. L. Divakar Allavarpu**[1] | **Vankamamidi S. Naresh**[2] | **A. Krishna Mohan**[1]

[1]Computer Science Engineering, UCEK, JNTUK, Kakinada, Andhra Pradesh, India

[2]Department of Computer Science and Engineering, Sri Vasavi Engineering College, Tadepalligudeam, Andhra Pradesh, India

**Correspondence**
Vankamamidi S. Naresh, Department of Computer Science and Engineering, Sri Vasavi Engineering College, Tadepalligudeam 534101, Andhra Pradesh, India.
Email: vsnaresh111@gmail.com

**Abstract**

With the growing significance of Credit Risk Analysis (CRA) with a focus on privacy, there is a pressing demand for a Privacy Preserving Machine Learning (PPML) decision support system. In this context, we introduce a framework for privacy-preserving credit risk analysis that utilizes Homomorphic Encryption aware Logistic Regression (HELR) on encrypted data. The implementation involves the use of TenSEAL and Torch libraries for Logistic Regression (LR), integrating the proposed HELR on polynomial degrees 3 and 5 across German, Taiwan, Japan, and Australian datasets. The presented model yields satisfactory results compared to non-Homomorphic Encryption (HE) models, demonstrating a minimal accuracy difference ranging from 0.5% to 7.8%. Notably, HELR_g5 outperforms HELR_g3, exhibiting a higher Area Under Curve (AUC) value. Additionally, a thorough security analysis indicates the resilience of the proposed system against various privacy attacks, including poison attacks, evasion attacks, member inference attacks, model inversion attacks, and model extraction attacks at different stages of machine learning. Finally, in the comparative analysis, we highlight that the proposed model ensures data privacy, encompassing training privacy and model privacy during the training phase, as well as input and output privacy during the inference phase a level of privacy not achieved by existing systems.

**KEYWORDS**

cloud service provider, credit risk analysis, homomorphic encryption, logistic regression, privacy-preserving machine learning

## 1 | INTRODUCTION

Credit risk is the possibility of financial losses if a borrower fails to fulfill their financial obligations. Although numerous factors contribute to credit risk, lenders can mitigate this risk by conducting due diligence during the loan granting process and continuously monitoring borrower payments and other behaviors to reduce the likelihood of accumulating Non-Performing Assets (NPAs) and fraud.

With the rise of big data, many organizations are outsourcing their data storage and processing to cloud-based services. As data has become an increasingly valuable resource, data holders must maintain complete privacy and control over

  

their data. Privacy-Preserving Machine Learning (PPML) is a technique designed to extract valuable insights from data while preserving its privacy.

One of the ways to address this challenge is to use HE, a method that allows computations to be performed on encrypted data without ever decrypting it. HE can process sensitive financial data, such as loan applications, while keeping the data private and secure. LR is a statistical model that can be used to analyze data and make predictions. Banks often uses it to analyze the risk associated with loan applications and make informed decisions about whether to approve or reject the loan.

Training and utilizing LR models on encrypted data is a feasible option by using HE. This means sensitive loan application data can be kept private and secure while allowing the bank to analyze the data and make informed decisions.

In this approach, the loan application data is encrypted using HE, and build the LR model on encrypted data. After the completion of model training, it becomes possible to utilize it for making predictions on new loan applications without ever decrypting the data. Finally, the predictions are decrypted using HE techniques to reveal whether the loan application should be approved or not. This approach ensures that sensitive bank loan application data remains private and secure throughout the process, allowing the bank to make informed decisions.

Privacy preservation of data could be done in various stages including training, input, output, and model privacy. Training privacy ensures the privacy of training data is imperative to prevent any malicious attempts at reverse engineering the training data. Input privacy, which seeks to uphold the confidentiality of data throughout both the training and inference phases. This is especially critical when data is sent to an untrusted third party, like a cloud server, for computational tasks.

The output data privacy refers to the protection and confidentiality of information revealed through the outputs or predictions generated by a model during its inference phase. In Model privacy, is designed to secure the non-disclosure of attributes defining a model, such as its architecture and weights. This ensures protection against malicious entities attempting to steal the model.

In this direction, this article proposes a Credit Risk Analysis (CRA) based on HE-aware LR in the cloud for providing data privacy at various stages of Machine Learning (ML), such as training, input, model, and output.

### 1.1 | Contributions

The main contribution of this work is to establish Privacy-Preserving Credit Risk Analysis (PPCRA) Framework with the following:

- Established a HE-aware Logistic Regression model is built on encrypted data using the CKKS encryption scheme.
- Established Security analysis that demonstrates the proposed system's ability to defend against various privacy attacks, including poisoning, evasion, member inference, model inversion, and model extraction.
- Experiments were conducted using the TenSEAL package on real-world financial datasets concerning Germany, Japan, Australia, and Taiwan.
- The performance analysis of LR over unencrypted data sets and encrypted data sets of the proposed system was evaluated.

The organization of the remaining article is as follows: the related work about Privacy-Preserving Logistic Regression (PPLR) was presented in Section 2. The background knowledge of logistic regression, homomorphic encryption, and polynomial approximation is introduced in Section 3. The CRA model and framework proposed in the article are described in Section 4. In Section 5, a security analysis is conducted, while Section 6 includes experiments that demonstrate the efficiency and accuracy of the approach. In Section 7 we concluded by providing a comprehensive summary of the key findings and outlines potential avenues for future research.

## 2 | RELATED WORK

There has been increasing interest in using HE for privacy-preserving data analysis, including bank loan processing based on LR. This section is dedicated to illustrates the recent development of PPLR-based credit risk evaluation systems.

Various risk models have been proposed in the literature for credit risk assessment. For instance, least squares support vector machines, fuzzy logistic-based approaches,[1] and imbalanced ensemble learning techniques that use regression ensembles to predict credit risk.[2] Furthermore, a post-processing approach that is agnostic and based on correlation network models was employed and has been suggested to measure creditworthiness for sound financial decision-making.[3] At the same time, data mining techniques have been applied to develop dynamic credit risk assessment models.[4]

Other models, such as artificial neural networks[5] and deep neural networks,[6] have also been proposed for predicting credit scores and assessing credit risk. While these models enable creditors to make quick credit decisions based on credit data obtained from bureaus, existing risk models fail to account for the privacy protection of user credit data and their associated weights. This implies that creditors typically acquire credit data for these models without sufficient regard for privacy protection.

As cloud computing has become more prevalent, several studies have focused on addressing security concerns related to this technology. For example, research has been conducted on enhancing the security of cloud frameworks,[7,8] ensuring location privacy in mobile cloud environments,[9,10] and improving security for cloud storage solutions.[11,12] It triggers the need for ML over encrypted data in the cloud.

Existing privacy-preserving techniques used in data mining[13–16] and machine learning techniques[17–19] are data perturbation and cryptography. Reference 20 proposed a blockchain-assisted method to enhance security using Attribute-based searchable encryption (ABSE) of fine-grained searchable encryption in a cloud-based healthcare cyber-physical system. This innovative solution aims to address critical security concerns related to sensitive healthcare data in cloud environments. Reference 21 proposed a novel approach to enhance the security of cloud data by introducing a unique encryption mechanism. This study provides insights into a hybrid encryption strategy for cloud computing environments, leveraging XOR and Genetic Algorithm for robust data protection.

Some of the researchers are working on various security mechanisms for various applications. Reference 22 proposed an approach to address the challenges of secure image sharing on social media platforms, offering a reliable solution for protecting digital content. Reference 21 introduced a novel four-image encryption scheme that is employed for image encryption, introducing complexity and security. It is incorporated for key generation, enhancing the encryption process. Additionally, computer-generated holograms contribute to the overall security of the proposed encryption scheme.

Many researchers are working on PPLR using HE. Aono et al.[23] presented an interactive, secure computation protocol that employs additive HE for computing LR. Their work demonstrated the effectiveness of HE in LR. Similarly, Kim et al.[24] utilized HE to assess the learning phase of LR on different biological datasets involved in using the gradient descent method. However, their approach is limited in handling large datasets for practical evaluation. Further, they have not demonstrated privacy preservation in various stages including input, output, training, and model privacy.

On the other hand, Mohassel and Zhang[25] developed Secure Multi-Party Computation (SMPC) protocols for LR but utilized garbled circuits instead of HE. Another relevant study was conducted by Wang et al.,[26] which introduced Homomorphic computation of ExAct Logistic rEgRession (HEALER), a system capable of computing precise LR models in a homomorphic manner. Moreover, their methodology is limited to a solitary attribute and a relatively small quantity of records, making it impractical for many applications. Furthermore, fewer iterations are only feasible as each of its iterations are computationally expensive.

While there has been considerable interest in privacy preserving CRA systems, limited academic research has been conducted directly related to this topic. Zheng et al.[27] introduced a Privacy-preserving Credit risk modeling based on Adversarial Learning (PCAL) modeling framework. PCAL is utilized to obscure private data while preserving essential information for the target prediction from the original dataset. PCAL's effectiveness in terms of utility and privacy protection as compared to off-the-shelf alternatives can help alleviate privacy concerns for clients of financial firms.

In addition, Khan et al.[28] proposed an ML model called Blind faith, which trains on plaintext data but performs user input classification on homomorphically encrypted ciphertexts which cannot provide training data privacy. The construction was designed in a particular way to ensure compatibility with HE. By conducting computations directly on encrypted data, Blind faith enables high-accuracy predictions while preserving users' privacy. However, the model's privacy is compromised even though the client's input is encrypted while the model is in plaintext.

Li et al.[29] proposed a homomorphic encryption framework over non-abelian rings and defined the homomorphism operations in ciphertexts space. This scheme can achieve one-way security and provides privacy preservation for machine learning training and classification in data ciphertexts environment. Chiang[30] proposed a faster gradient variant called quadratic gradient for privacy preserving logistic regression training to improve the performance of the training. However, these methods must refresh the ciphertexts by different updates were time consuming. However, these methods do not provide demonstrations about privacy in various stages.

From the above limitations of the existing CRA it is essential to build a Privacy-Preserving CRA arises from the need to balance the critical task of assessing individuals' creditworthiness with the protection of their sensitive personal information. The privacy preserving CRA is rooted in the need to balance the imperative of accurate risk assessment with the ethical and legal obligation to protect individuals' privacy, comply with regulations, build and maintain consumer trust, and enhance overall cybersecurity in the financial sector.

## 3 | BACKGROUND KNOWLEDGE

This section provides a brief background of LR and HE for the proposed system.

### 3.1 | Notations

Table 1 shows the list of symbols and their denotation used in the subsequent sections.

### 3.2 | Logistic regression and gradient descent method

Logistic regression (LR) is a supervised learning technique for predicting the probability of an event. An LR model is trained by optimizing coefficients that are applied to a feature vector as a linear combination. Suppose $D$ represents the number of features and $N$ represents the number of training samples. Let $(X, Y) = \left\{\left(x_{ij} y_i\right) | i = 1, \ldots, n, \text{ and } i = 1, \ldots, m\right\}$ be the training data, where $X \in R^{n \times m}$ and $Y \in R^{1 \times m}, x_{ij} = (x_{i1}, \ldots, x_{in})$ denotes the input features and $y_i \in \{0, 1\}$ denotes the corresponding output will optimize the weight parameters $\omega \in R^{(n \times 1)}$. LR

**TABLE 1** Symbols and notations.

| Symbol | Denotation |
|---|---|
| $n$ | Polynomial modulus degree (ring dimension) |
| $m$ | Raw message $\in C^{n/2}$ is a message vector of Gaussian integers of size $n/2$ |
| $p$ | Plaintext coefficient modulus |
| $q$ | Ciphertext coefficient modulus |
| $\lambda$ | Encryption security parameters $(n, p, q)$ |
| $R = \mathbb{Z}[X]/(X^n + 1)$ | Cyclotomic polynomial rings |
| $R_p = \mathbb{Z}_p[X]/(X^n + 1)$ | Plaintext polynomial ring |
| $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ | Ciphertext polynomial ring |
| $(pk, sk)$ | The public key and private key pair |
| $\Delta$ | Scaling factor |
| $\tau(\mu)$ | Plaintext polynomial $\tau(\mu) \in R_p$ (i.e., Encoded vector) |
| $e$ | Error polynomial |
| $N$ | Number of samples in the dataset |
| $X$ | Plain text vector input |
| $Y$ | Plain text vector output |
| $\omega$ | Plain text vector weights |
| $[X]$ | Encrypted vector input |
| $[Y]$ | Encrypted vector output |
| $[\omega]$ | Encrypted vector weights |

can be generalized from linear regression by mapping the whole real line $a = \sigma\left(x_i\omega^T\right)$ to (0, 1) via the sigmoid function $\sigma(y) = \frac{1}{(1+\exp(-y))}$. Thus, the LR can be formulated with the class label $y \in \{0, 1\}$ as follows:

$$\Pr(y_i = 0|x_i) = \sigma\left(x_i\omega^T\right) = \frac{1}{\left(1 + e^{-x_i\omega^T}\right)},$$
$$\Pr(y_i = 1|x_i) = \sigma\left(x_i\omega^T\right) = \frac{1}{\left(1 + e^{-x_i\omega^T}\right)}.$$

LR sets a threshold (usually 0.5) and compares its output with it to decide the resulting class label. To obtain the optimized weight parameters $\omega$, we must minimize the loss function, which can be attained through the use of gradient descent method. The loss function is calculated using the following formula:

$$L(\omega) = \frac{1}{n}\log\sum_{i=1}^{n}\left[y_i \log a + (1 - y_i)\log(1 - a)\right], \text{for } i = 1, \ldots, n.$$

To minimize the loss function using gradian decent, we use Algorithm 1. The objective of the gradient descent calculation is to determine the minimum value of the loss function in the direction of gradient descent. In each iteration, the gradient decreases and at the $t^{th}$ iteration, the weight vector $\omega \in R^n$ is updated as follows:

$$\omega_t = \omega_{t-1} - \alpha\, X^T\left(\sigma\left(X\,\omega_{t-1}^T\right) - y\right),$$

where, $\alpha$ denotes the learning rate.

**Algorithm 1.** GradiantDecentLR ($X$, $Y$, $\boldsymbol{\alpha}$)

Input: X: Training input vector $\in R^{ny}$, Y: Output vector $\in R^n$, $\alpha$: Learning rate >0
Output: $\omega$: Weight vector $\in R^n$

Initialize weight vector: $\omega = 0$
for $iter = 0$ to t do
  for $i = 0$ to $n$ do
    $p_i = (x_i * \omega^T)$
    $q_i = \sigma\,(p_i)$
  end
  for $j = 0$ to $m$ do
    $r_j = \sum_i (q_i - y_i)x_{ij}$
    $\omega_j = \omega_j - \alpha * r_j$
  end
end
return $\omega$

## 3.3 | Fully homomorphic encryption

HE is a cryptographic technique that enables an application to perform computations on encrypted data instead of the original data itself. This allows data owners to keep their information encrypted while allowing a third party to perform computations without knowing the underlying data. The encrypted results can be sent back to the data owner, who is the only one with the ability to decrypt the data. Traditional encryption methods only allow for the storage and retrieval of data, but with HE, operations can be performed on the encrypted data without Decryption. This provides a more efficient and secure alternative to traditional encryption schemes. In 2009, Craig Gentry[31] introduced the concept of bootstrapping, which led to the development of Fully Homomorphic Encryption (FHE) schemes based on lattices and the Learning With Errors problem (LWE).

We adopted the CKKS (Cheon, Kim, Kim, Song)[32] is a FHE scheme designed for efficiently handling computations on encrypted data with a cryptographic approach that allows computations on encrypted data involving real-number arithmetic. FHE allows computations to be performed on encrypted data without the need for decryption, providing a powerful tool for privacy-preserving computation in various applications.

Key features of CKKS FHE scheme include:

1. *Homomorphic Operations*: CKKS supports both addition and multiplication operations on encrypted data. This enables a wide range of mathematical operations to be performed on encrypted inputs.
2. *Efficient Polynomial Approximations*: CKKS leverages polynomial approximations to handle the mathematical operations efficiently. This is crucial for practical usability, as fully homomorphic encryption schemes often involve complex mathematical computations.
3. *Parameter Selection*: The CKKS scheme allows for flexibility in parameter selection, enabling users to adjust security levels and efficiency based on their specific requirements. This adaptability makes it suitable for various use cases.
4. *Applications*: CKKS is particularly well-suited for applications involving real-number arithmetic, making it useful for tasks such as machine learning on encrypted data, secure outsourcing of computations, and privacy-preserving data analysis.

Overall, CKKS is a significant advancement in fully homomorphic encryption, providing a balance between security and efficiency, which is essential for practical implementation in real-world scenarios.

### 3.4 | Polynomial approximation of the sigmoid function

One of the limitations of the current HE cryptosystems is their inability to support operations beyond polynomial arithmetic. This poses a challenge when implementing LR, as the evaluation of the sigmoid function cannot be expressed as a polynomial. Traditional methods for finding polynomial approximations, such as Taylor expansion and Lagrange interpolation, provide accurate approximations only within a small range close to the point of interest, with approximation errors significantly increasing outside that range. To address this issue, a global polynomial approximation of the sigmoid function, Kim et al.[33] employed the least squares approach that provides a good approximation on an extensive range. In this study, we have utilized this approximation method and computed the degree 3 and 5 least squares polynomials of the sigmoid function over $[-5, 5]$ domain., which are represented by the approximate polynomials $g(x)$ as follows:

$$\begin{cases} g_3(x) = 0.5 - 1.20096 \cdot (x/8) + 0.81562 \cdot (x/8)^3 \\ g_5(x) = 0.5 - 1.20096 \cdot (x/8) + 0.81562 + \\ \qquad 2.3533056 \cdot (x/8)^3 - 1.3511295 \cdot (x/8)^5. \end{cases}$$

The concept of Least Squares Fitting Polynomial involves finding a polynomial of degree $d$ that has the smallest square sum error, $(p(y_i)\text{-}p(x_i))^2$, among all polynomials of degree $\leq d$, given $N$ points $(x_i, y_i)$. However, the current HE schemes can only handle polynomial arithmetic computations, making the computation of the sigmoid function $\sigma(x) = \frac{1}{(1+\exp(-x))}$ A challenging task. To overcome this obstacle, we aim to use low-degree polynomials requiring fewer evaluation depths while maintaining high precision. Our study considers the use of the least squares approach to approximate the sigmoid function with polynomials of degree 3 and 5, denoted as $g_3(x)$ and $g_5(x)$, respectively. The maximum errors between $\sigma(x) = \frac{1}{(1+\exp(-x))}$ and the least squares polynomials $g_3(x)$ and $g_5(x)$ are approximately 0.114 and 0.061, respectively.

## 4 | PROPOSED SYSTEM

### 4.1 | System model

The system model of the proposed Credit Risk Analysis (CRA) considers three entities, namely a Customer (C), Bank (B), and Cloud Service Provider (CSP), shown in Figure 1.

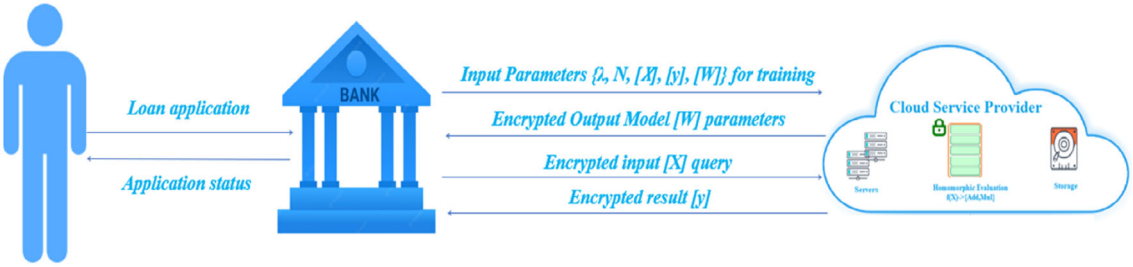


**FIGURE 1** System model for privacy-preserving credit risk analysis (PPCRA).

Cloud Service Provider (CSP) is semi-trusted and can offer virtually limitless storage and computation resources to users as services over the internet. These services can be utilized to operate on encrypted data while concealing platform and implementation particulars from users.

Customer (C): Who is a person or organization who seeks a loan from the bank for their financial requirement. For this, they must apply for a loan from the bank. The application consists of customer personal sensitive information to avoid privacy breaches in the public cloud, the user encrypts their sensitive data before being outsourced to the public cloud.

Bank (B) is an institution that deals in money and loans to customers based on their credit history. Bank has limited storage and computational resources and wants to use CSP for data analysis service on encrypted data to train an LR model without revealing its own training data privacy and stores the encrypted train data into CSP. Initially, the bank trains the encrypted LR model and supports customer querying and extracting the results to accept or reject the loan.

## 4.2 | Privacy-preserving credit risk analysis framework (PPCRAF)

In this section, we first discuss a Privacy-Preserving Credit Risk Analysis Framework (PPCRAF), and various phases associated with the algorithms of this framework.

The PPCRA is a three phases framework consisting of

i. Privacy-Preserving Training (PPT).
ii. Privacy-Preserving Prediction Query (PPPQ).
iii. Privacy-Preserving Result Extraction (PPRE).

We depicted CRAF processing in the above phases in Figure 2.

### 4.2.1 | Privacy-preserving training for credit risk analysis

This phase is intended for a training classifier built over encrypted Credit Risk Analysis (CRA) data owned by a bank. This data is sent to the cloud data center for establishing an evaluation model, which includes various sub-phases as follows:

*Initialization*

At the beginning, a setup process is executed to initialize the Homomorphic Encryption for Arithmetic of Approximate Numbers (HENNA) scheme and distribute the security parameters of the system $\lambda$ to generate the public key pair. Let us consider $n$ to be the dimension of the ring (power of two), cyclotomic polynomial rings $R = \mathbb{Z}[X]/(X^n + 1)$, plain text polynomial: $R_p = \mathbb{Z}_p[X]/(X^n + 1)$, and cipher text polynomial: $R_q = \mathbb{Z}_q[X]/(X^n + 1)$.

*Key generation*

In this phase, bank (B) produces a pair of keys $(pk_b, sk_b)$ by invoking KeyGeneration() Algorithm 2.

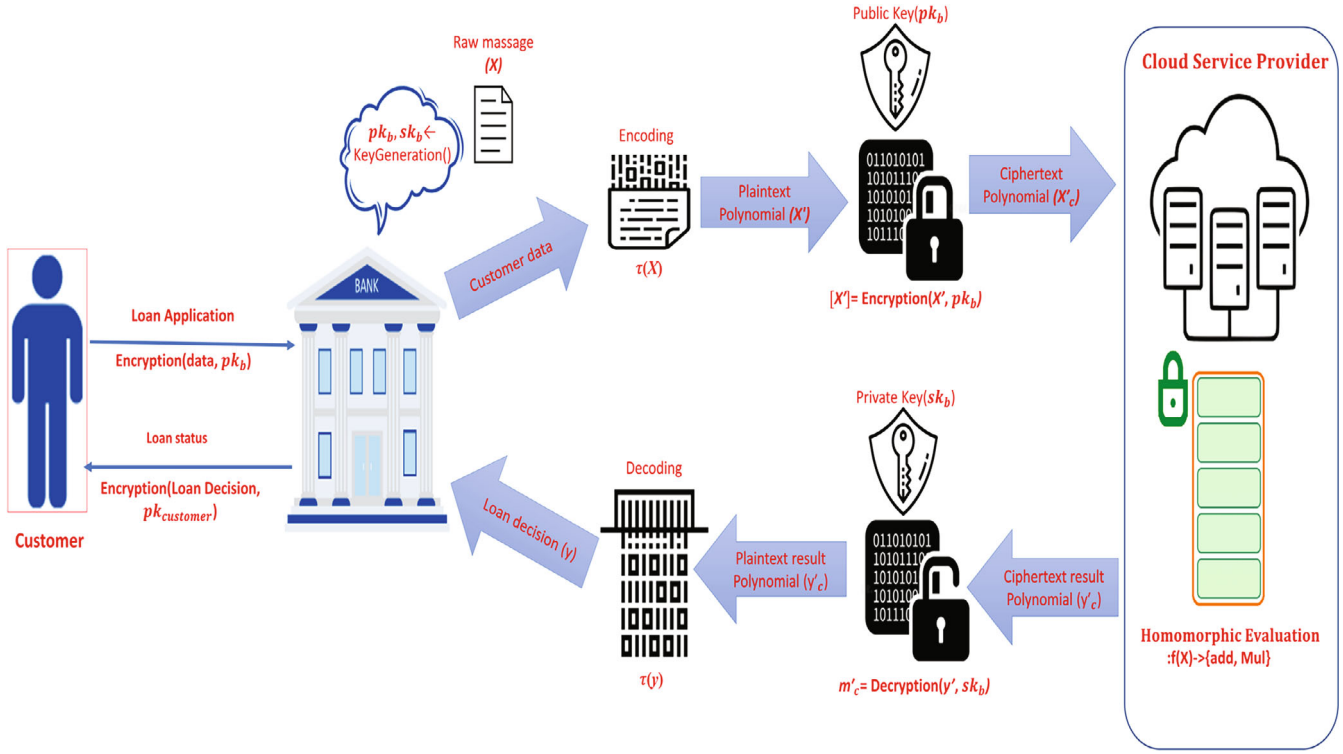


**FIGURE 2** Privacy-preserving credit risk analysis framework (PPCRAF).

**Algorithm 2.** KeyGeneration (λ)

Input: $\lambda$: security parameter
Output: $pk$: polynomial public key pair $(pk_1, pk_2)$, $sk$: secret key,

Select *'a'* is a random polynomial sampled uniformly from $R_{qL}$.
Select *'e'* is a random error polynomial.
Choose a secret key L'*sk*' $\in R_{qL}$
Compute polynomial public Key pair $pk \leftarrow (pk_1, pk_2)$ as:

$pk_1 = [-a.sk + e] \; mod \; q_L$

$pk_2 = a$

return $(pk, sk)$

*Encoding*

In this phase, the bank encodes sensitive message input vector $X_i \in C^{n/2}$ of Gaussian integers of size $n/2$ into a plaintext polynomial and the associated desired output vector $y_o \in C^{n/2}$ as $(m_i, m_o) \in m$. All the collected data, that is, message vector, is encoded into a plaintext polynomial using Algorithm 3 to support approximate arithmetic over real numbers efficiently. Further, the scaling factor (Δ) lets us control the encoding/decoding error in the rounding process.

**Algorithm 3.** Encode ($m$, Δ)

Input: $m$: message vector $m \in C^{n/2}$ of size $n/2$; Δ: Scaling Factor
Output: $\tau(\mu)$: plaintext polynomial $\in R$.

For given input a message $m \in C^{n/2}$ which is a vector of Gaussian integers

Let $\tau: K = \mathbb{Q}[X]/(X^n + 1) \rightarrow C^{n/2}$

Let $\tau(a) = (a(\zeta), a(\zeta 5), ..., a(\zeta 2n\text{-}3))$ where $\zeta = \exp(\pi_i/n)$.

1. $a(X) = \tau^{-1}(m)$
2. $\mu(X) = \Delta . a(X)$
3. $\tau(\mu) = (\mu(\zeta), \mu(\zeta 5), ..., \mu(\zeta 2n\text{-}3))$

return $\tau(\mu)$

*Encryption*

In this phase, the plaintext polynomial $(m_i, m_o) \in m$ will be encrypted with bank's public key $pk_b$ using Algorithm 4 to generate cipher text pair $(c_1, c_2)$ outsourced to the cloud server $S$ for building the encrypted LR model.

**Algorithm 4.** Encryption $(m, pk)$

Input: $m$: Encoded Plain Text message, $pk$: public key
Output: $C$: Cipher text pair $(c_1, c_2)$, $sk$: secret key,

Generate '$u$' is a random polynomial sampled uniformly from $R_2$.
Select '$e_0, e_1$' is a random error from discrete Gaussian distribution.
Choose a secret key '$sk$' $\in R_{qL}$
Compute polynomial public Key pair $pk \leftarrow (pk_1, pk_2)$ as:

$$c_1 = \left[u.pk_1 + e_1 + m\right] \bmod q_L$$
$$c_2 = \left[u.pk_2 + e_2 + m\right] \bmod q_L$$

return $C = (c_1, c_2)$

*Encrypted training*

Once the cloud server $S$ receives the encrypted training data set $\chi = \left\{\lceil x_i \rceil_{pkb}, \lceil y_i \rceil_{pkb}\right\}$ where $1 <= i <= N$. Here $y_i$ denotes the desired binary class label {0, 1} for rejecting or accepting the credit. Further, the cloud server is required to create an optimized approximation of $\lceil y_i \rceil_{pkb}$ by using the classifier model $\varphi\left(\lceil x_i \rceil_{pkb}, \lceil y_i \rceil_{pkb}\right)$. The classifier model, denoted by $\varphi(\chi, \theta)$ specifies the hypothesis class and a specific value of parameter $\theta$ instantiates one hypothesis from the hypothesis class.

In this model training on encrypted data using Algorithm 5. The cloud server computes the approximation value for the sigmoid function of the LR denotes as $\varphi(.)$. Cloud server computes the approximation value $\varphi\left(\lceil x_i \rceil_{pkb}, \lceil y_i \rceil_{pkb}\right)$, then it is necessary to calculate the difference between the intended vector $\lceil y_i \rceil_{pkb}$ and the estimated value using Algorithm 4.

**Algorithm 5.** GradiantDecentLREncrypted ($[X]$, $[y]$, $[\boldsymbol{\omega}]$, $\boldsymbol{\alpha}$)

Input: [X]: Encrypted training input vector $\in R^{nm}$, [y]: Encrypted Output vector $\in R^n \in$ Rn, $\alpha$: Learning rate >0
Output: $[\omega]$: encrypted Weight vector $\in R^m$

```
for j = 0 to t do
  for i = 0 to n do
    [p_i] = σ([x_i] * [ω]^T)
    [q_i] = sub([p_i], [y_i])
  end
  for j = 0 to m do
      [q_j] = Σ_i([p_i] − [y_i])[x_ij]
      [ω_j] = [ω_j] − α*[q_j]
  end
end
return [ω]
```

*Addition and multiplication operation in HE*

When working with an encrypted domain, secure addition using Algorithm 6 and multiplication Algorithm 7 can be used to perform computations directly due to the FHE of the scheme. However, if the discriminant function $G$ is non-linear, it may be approximated using alternative methods such as linear functions, or by employing Taylor or Maclaurin series.

**Algorithm 6.** Addition $(\boldsymbol{C^1}, \boldsymbol{C^2})$

Input: $C^1$: Cipher text pair $(C_1^1, C_2^1)$, $C^2$: Cipher text pair $(C_1^2, C_2^2)$
Output: $C^3$: Cipher text pair $(C_1^3, C_2^3)$

$C_1^3 = (C_1^1 + C_1^2) \bmod q_l$
$C_2^3 = (C_2^1 + C_2^2) \bmod q_l$
$C^3 \leftarrow (C_1^3, C_2^3)$
return $C^3$

**Algorithm 7.** Multiplication $(\boldsymbol{C^1}, \boldsymbol{C^2})$

Input: $C^1$: Cipher text pair $(C_1^1, C_2^1)$, $C^2$: Cipher text pair $(C_1^2, C_2^2)$
Output: $C^3$: Cipher text pair $(C_1^3, C_2^3, C_3^3)$

$C_1^3 = (C_1^1 \cdot C_1^2) \bmod q_l$
$C_2^3 == (C_1^1 \cdot \ C_2^2) + (C_2^1 \cdot \ C_1^2) \bmod q_l$
$C_3^3 = (C_2^1 \cdot \ C_2^2) \bmod q_l$
$C^3 \leftarrow \left(C_1^3, C_2^3, C_3^3\right)$
*return* $C^3$

*Decoding*

After completing the training, the cloud server sends the encrypted model $\varphi(.)$ to the bank. The bank decodes the model using Algorithm 8 and passes it to the next phase.

**Algorithm 8.** Decode $(P, \Delta)$

Input: p: plaintext polynomial; $\Delta$: Scaling Factor
Output: m: message vector $\in C^{n/2}$

For given plaintext polynomial $P \in R$
Let $\tau$: $K = \mathbb{Q}[X]/(X^n + 1) \rightarrow C^{n/2}$
Let $\tau(a) = (a(\zeta), \ a(\zeta 5), ..., a(\zeta 2n\text{-}3))$ where $\zeta = \exp(\pi_i/n)$
1. $a'(X) = \tau(p)$
2. X$\mu'(X) = \Delta'.a'(X)$
return $\mu'(X)$

*Decryption*

Finally, the bank decrypts the model parameters with its private key $sk_b$ and gets the plaintext model using Algorithm 9, the model trained from the local private data as in Algorithm 5.

**Algorithm 9.** Decryption $(c, sk)$

Input: C: Cipher text pair $(c_1, c_2)$, sk: secret key,
*Output*: $m$: Encoded Plain Text message

$m = \left[c_1 + c_2 \cdot sk\right] \; mod \; q_L]$
return $m$

### 4.2.2 | Privacy-preserving prediction query (PPPQ)

Once a classification model $\varphi(.)$ has been learned over an encrypted domain using the parameter $\theta^*$, the cloud server can run this classifier to a new encrypted feature vector $[z]_{pkb}$ sent by the bank. In this scenario, if the bank queries $z$ with $m$ attributes $z = (z_1, z_1, \ldots, z_m)$, and encrypts the $[z]_{pkb} \leftarrow \text{Encrypt}\left(z, pk_{pkb}\right)$ generates encrypted data query $[z]_{pkb} = \left([z_1]_{pkb}, [z_2]_{pkb}, \ldots, [z_m]_{pkb}\right)$ the cloud server can use the learned model to make predictions over the encrypted data. Now the cloud server runs the classifier $\varphi(.)$ over feature vector $[\mathbf{z}]_{pkb}$ and uses the encrypted parameter $[\theta^*]_{pkb}$ to generate the prediction $[y]_{pkb} = \varphi\left([z]_{pkb}, [\theta^*]_{pkb}\right)$.

### 4.2.3 | Privacy-preserving result extraction (PPRE)

After computing the transformed prediction results $[y]_{pkb}$, the cloud server sends them back to the bank. Upon receiving the encrypted prediction $[y]_{pkb}$, the bank can decrypt it using the Decryption algorithm. This involves using the bank's private key $sk_b$ and a decryption function to obtain the original prediction using cipher text $[y]_{pkb}$ as input and outputs $y \leftarrow \text{Decrypt}\left(sk_{pkb}, [y]_{pkb}\right)$. Later, based on the classification status, it will inform the customer whether the credit is approved or rejected in an encrypted format.

## 5 | SECURITY ANALYSIS

In this section, we presented mathematical models for various privacy attacks on LR and HE-based solutions to defend against these attacks. First, we have introduced LR and various possible privacy attacks on LR over financial data are visualized in Figure 3.

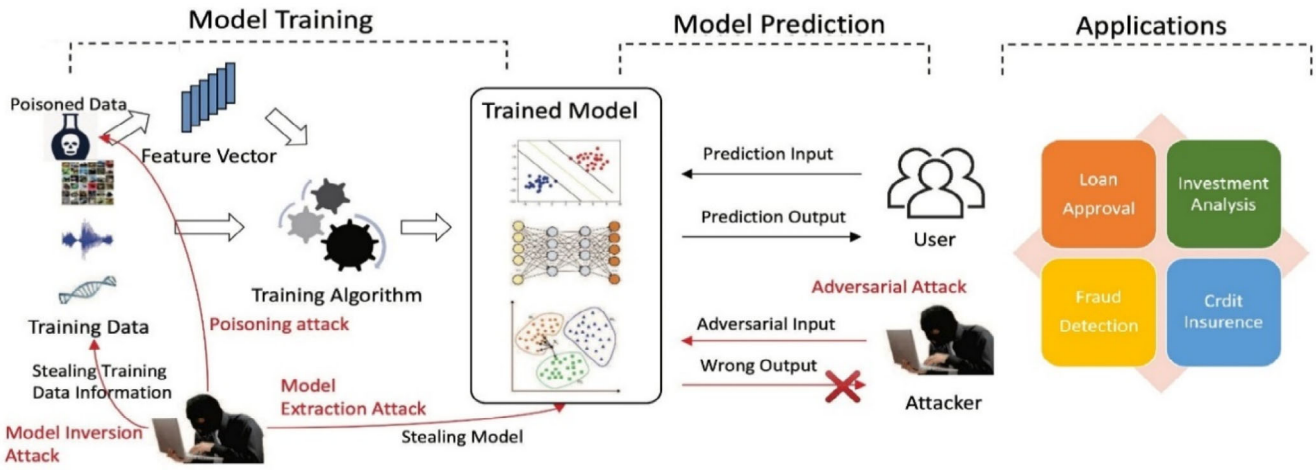


**FIGURE 3** Privacy attacks on logistic regression.

The logistic regression model can be defined as follows:

First, let $X$ be an $n \times m$ matrix representing the training set of $n$ data points, each with $m$ features. Let $Y$ be an $n$-dimensional column vector representing the corresponding binary labels of the data points, where each element of $Y$ is either 0 or 1, where the goal is to predict the label $y$ for a given input vector $x$.

$$p(y_i = 0|x_i) = \sigma\left(x_i \omega^T\right),$$

$$p(y_i = 1|x_i) = \sigma\left(x_i \omega^T\right),$$

where, $\sigma(z) = 1/(1 + \exp(-z))$ is the sigmoid function and $\omega$ is the weight vector learned during training.

## 5.1 | A mathematical model for poisoning attacks on logistic regression

Let us assume that the attacker has access to the training dataset $\{(x_1 y_1), (x_2 y_2), \ldots, (x_n y_n)\}$ used to train the LR model. The attacker's goal is to modify a subset of the training data to maximize the model's loss function. The loss function for LR can be defined as follows:

$$L(\omega) = -\sum_i y_i \ \log\left(\sigma\left(w^T x_i\right)\right) + (1 - y_i) \ \log\left(1 - \sigma\left(w^T x_i\right)\right). \tag{1}$$

The attacker can introduce a set of malicious examples $\left\{x'_m, y'_m\right\}$ into the training dataset or perturb existing examples $\{x_i, y_i\}$ to create new examples $\left\{x'_i, y'_i\right\}$. The perturbed examples can be generated by adding a small perturbation $\Delta x_i$ to the original input $x_i$, such that:

$$x'_i = x_i + \Delta x_i.$$

The attacker's objective is to find a set of perturbed examples that will cause the model to misclassify certain examples during testing. This can be achieved by solving the following optimization problem:

- maximize $L(\omega)$;
- subject to $y'_i = y'_i$ for a subset of the examples;
- where $y'_i$ is the label assigned to the perturbed example $x'_i$.

### 5.1.1 | HE-based solution to defend against poisoning attack

The LR model aims to learn the weights, w, that minimize the logistic loss function:

$$L(\omega) = -1/n \ \sum_i \left(y_i \ \log(p_i) + (1 - y_i) \log(1 - p_i)\right),$$

where, $p_i = 1/\left(1 + \exp\left(-x_i \omega^T\right)\right)$ is the predicted probability of the $i$th data point belonging to the positive class.

Assuming that the training data and labels are encrypted using an FHE scheme, we can perform computations on the encrypted data without decrypting it.

To train the LR model, we need to compute the gradient of the logistic loss function with respect to the weights, $w$. This gradient can be expressed as:

$$\text{Grad } L(\omega) = 1/n \ \sum_i ((p_i - y_i) x_i).$$

To compute this gradient on the encrypted data, we can use an FHE scheme that supports arithmetic operations such as addition and multiplication using the following steps:

1. Encrypt the training data matrix $X$ and the label vector $Y$ using the Encryption() function in the FHE scheme so that we have $[X]$ and $[Y]$.
2. Initialize the weights vector, $\omega$, to a random value.
3. Encrypt the weights vector using the FHE scheme so that we have encrypted weight vector $[\omega]$.
4. For each training iteration:
   a. Compute the predicted probability vector $P$ using the encrypted weights and encrypted data as follows:

$$[P] = 1/\left(1 + \exp\ \left(-[X] \cdot \left[\omega^T\right]\right)\right).$$

   b. Compute the encrypted gradient vector, [Grad $L(\omega)$], as follows:
      - [Grad $L(\omega)$] $= 1/n \sum([P] - [Y] \cdot [X])$.
   c. Decrypt the encrypted gradient vector using Decryption() function to obtain the actual gradient vector Grad $L(\omega)$.
   d. Update the weights vector, $\omega$, using the decrypted gradient vector.
   e. Encrypt the updated weights vector using the FHE scheme.

After a sufficient number of training iterations, the final weights vector, $[\omega]$, can be decrypted using Decryption() function to obtain the trained LR model $\omega$.

Since the computations are performed on the encrypted data and the weights are encrypted throughout the training process, an attacker cannot modify the training data or the weights without having the decryption key, which should be kept secure. Therefore, HE can be used as a defense against poison attacks on LR.

## 5.2 | A mathematical model for evasion attacks on logistic regression

In this attack, the attacker modifies the input data to make it look legitimate but mislead the model's predictions.

Let us assume that the attacker has access to the trained model and the testing dataset $\left\{\left(x'_{test1}y'_{test1}\right), \left(x'_{test2}y'_{test2}\right), \ldots, \left(x'_{testn}y'_{testn}\right)\right\}$. The attacker's goal is to modify some of the input vectors in the test dataset to mislead the model's predictions.

The attacker can modify the input vector $x'_i$ by adding a small perturbation $\Delta x_i$ to it, such that:

$$x'_i = x_i + \Delta x_i.$$

The perturbation $\Delta x_i$ is chosen to minimize the distance between the original input $x_i$ and the perturbed input $x'_i$. This can be formulated as an optimization problem:

$$Minimize \left\|\Delta x_i\right\| \text{ subject to } f\left(x'_i\right) \neq y_i,$$

where, $f\left(x'_i\right)$ is the predicted label for the perturbed input $x'_i$. The objective is to minimize the magnitude of the perturbation while ensuring that the model misclassifies the perturbed input.

### 5.2.1 | HE-based solution for defending against evasion attacks on logistic regression

We need to show that the homomorphic operations performed on the encrypted data do not affect the accuracy of the LR model. This indicates that even if an attacker tries to modify the encrypted input data, the resulting decrypted output is still an accurate prediction of the original LR model.

Assuming that an attacker tries to modify the encrypted input vector $[X]$ by adding a perturbation vector Enc($p$), the new encrypted input becomes:

$$[X_{new}] = [X] + [P].$$

The attacker's goal is to make the decrypted output $Y_{new}$ different from the original output $Y$, by carefully choosing the perturbation vector $[P]$.

Let us assume that a maximum norm bounds the attacker's perturbation vector $||[P]|| \leq \epsilon$, where $\epsilon$ is a small value. This assumption is common in evasion attacks, where the attacker is limited in the modification they can make to the input data.

To show that the HE can preserve the accuracy of the LR model in the presence of such attacks, we need to prove that the difference between the decrypted outputs $y$ and $Y_{new}$ is bounded by a small value, even after the perturbation vector is added to the encrypted input as follows:

1. We can start by writing the original LR output $Y$ as:

$$Y = \sigma\left(\omega^T X + b\right).$$

2. We can also write the modified LR output $y_{new}$ as follows:

$$Y_{new} = \sigma\left(\omega^T X + b\right) = \sigma\left(\omega^T (X + P) + b\right).$$

3. Using the linearity of the dot product and the properties of the sigmoid function ($\sigma$), we can expand $Y_{new}$ as follows:

$$Y_{new} = \sigma\left(\omega^T X + b\right) + \omega^T P\, \sigma'\left(\omega^T X + b\right)\left(1 - \sigma\left(\omega^T X + b\right)\right).$$

Here, $\sigma'$ denotes the derivative of the sigmoid function. The second term in the above equation represents the perturbation introduced by the attacker's modification.

4. We can bound the absolute difference between $y$ and $Y_{new}$ as follows:

$$|\, Y_{new} - Y| = \left|\, \omega^T P\, \sigma'\left(\omega^T X + b\right)\left(1 - \sigma\left(\omega^T X + b\right)\right)\right|.$$

Using the fact that the sigmoid function is bounded between 0 and 1, we can obtain an upper bound on the above expression as follows:

$$|\, Y_{new} - Y| \leq ||\omega||||P||\ \ \left|\left|\sigma'\right|\right|_\infty.$$

Here, $||\omega||$ represents the norm of the weight vector $\omega$, and $||\sigma'||_\infty$ represents the maximum absolute value of the derivative of the sigmoid function. Since the perturbation vector is bounded by $||[P]|| \leq \epsilon$, we can obtain a tighter upper bound on the absolute difference as follows:

$$|\, Y_{new} - Y| \leq ||\omega|| \cdot \epsilon \cdot \left|\left|\sigma'\right|\right|_\infty.$$

This proves that the absolute difference between the decrypted outputs $Y$ and $Y_{new}$ is bounded by a small value that depends on the norm of the weight vector and the maximum absolute value of the derivative of the sigmoid function. By selecting appropriate values of these parameters and ensuring that the perturbation vector is small enough, we can preserve the accuracy of the LR model even in the presence of evasion attacks.

## 5.3 | A mathematical model for member inference attacks on logistic regression

An attacker can infer the presence of a specific individual in a trained LR model. In this attack, the attacker aims to determine whether a specific individual was present in the training dataset used to train the model.

Let us assume that the attacker has access to the trained model and a set of queries $Q = \left\{x_1', x_2', \ldots, x_m'\right\}$ where each query represents a specific individual's information. The attacker's goal is to determine whether a specific individual was present in the training dataset.

### 5.3.1 | HE-based solution for defending against member inference attacks on logistic regression

To defend against membership inference attacks using HE, we can encrypt both the training data and the weights using FHE. It allows computations to be performed on encrypted data without revealing the underlying plaintext.

1. We can now use the encrypted training data to perform gradient descent on the encrypted weights. The gradient descent update rule is given by:

$$\left[\omega'\right] = \left[\omega^T\right] - \alpha \cdot [X] \cdot \left([Y] - \left[Y'\right]\right).$$

2. Where $\alpha$ is the learning rate, $[X]$ is the encrypted input matrix, $\left[Y'\right]$ is the predicted output values computed using the current encrypted weights as follows:

$$\left[Y'\right] = \sigma\left(\left[\omega^T\right][X]\right).$$

3. After a fixed number of iterations of gradient descent, we obtain the final encrypted weights $[\omega^*]$ which can be used to predict the output values for new encrypted test data as follows:

$$\left[Y_{pred}\right] = \sigma\left(\left[\omega^*\right][X_{test}]\right).$$

4. To decrypt the predicted output values, we apply the decryption function as follows:

$$Y_{pred} = \text{Decryption}\left(\left[Y_{pred}\right]\right).$$

Since all the computations were performed on encrypted data, the attacker cannot determine if a specific data point was used to train the model. However, the computations can be computationally expensive due to the use of FHE, and the performance may be limited for large-scale datasets or models.

However, HE can be computationally expensive and may not be practical for large-scale datasets or models. Another potential limitation of HE is that it may not be suitable for all types of machine learning models. Therefore, it is essential to carefully consider the trade-offs between privacy and performance when deciding whether to use HE as a defense against membership inference attacks on LR.

## 5.4 | A mathematical model for model inversion attacks on logistic regression

An attacker can reconstruct the training data used to train an LR model. The attacker aims to infer sensitive information from the model's weights and predictions in this attack.

Let us consider a binary classification problem where the LR model is trained on a dataset $\{(x_1y_1), (x_2y_2), \ldots, (x_ny_n)\}$, where $x_i$ is the input vector and $y_i$ is the true label.

Let us assume that the attacker has access to the trained model and the model's predictions on a set of inputs $\left\{x'_1, x'_2, \ldots, x'_m\right\}$. The attacker's goal is to reconstruct the training data used to train the model. The attacker can use the LR model to calculate the partial derivative of the output with respect to each input feature $x_i$. This can be expressed as:

$$\partial p(y = 1|x, \omega)/\partial x_i = \sigma\left(\omega^T x\right)\left(1 - \sigma\left(\omega^T x\right)\right) * \omega_i.$$

The attacker can then use this information to reconstruct the training data. Specifically, the attacker can solve the following optimization problem:

- Minimize $\left\| x_i - x'_i \right\|$;
- subject to $p(y = 1|x_i, \omega) = p\left(y = 1|x'_i, \omega\right)$.

Where, $x_i$ is the original input vector used to train the model, $x'_i$ is the reconstructed input vector, and $p\left(y = 1|x'_i, \omega\right)$ is the predicted label for $x_i$ using the trained model. The objective is to find an input vector $x'_i$ that has the same predicted label as $x_i$ while being as close as possible to $x_i$.

### 5.4.1 | HE-based solution to defend against model inversion attacks

We use HE to protect the weights vector, $\omega$, of the LR model. We encrypt $\omega$ using an FHE scheme to obtain encrypted weight $[\omega]$.

To compute the predicted probability vector using HE, we first encrypt the input data point $x$ using the same FHE scheme to obtain $[X]$. We then compute the encrypted predicted probability $[P]$ as:

$$[P] = 1/\left(1 + \exp\left(-\omega^T x\right)\right).$$

We decrypt the encrypted predicted probability to obtain the actually predicted probability:

$$P = \text{Decryption}([P]).$$

We also add noise to the encrypted predicted probability $[P]$ which adds random noise to $[P]$ while maintaining its statistical properties to obtain $\left[P_{noisy}\right]$ is the encrypted predicted probability with added noise. We then encrypt a threshold value, $t$ to obtain $[t]$.

We compare $\left[P_{noisy}\right]$ to $[t]$, if $\left[P_{noisy}\right]$ is greater than or equal to $[t]$, we classify the corresponding data point as belonging to the positive class. Otherwise, we classify it as belonging to the negative class.

By encrypting the weights vector and adding noise to the predicted probability vector, this model provides a defense mechanism against model inversion attacks on LR. The encrypted weights vector prevents an attacker from directly obtaining the model parameters, while the added noise to the predicted probability vector prevents the attacker from accurately reconstructing the input data point.

## 5.5 | A mathematical model for model extraction attacks on logistic regression

It can help us understand how an attacker can extract a copy of a trained LR model without access to the original training dataset. In this attack, the attacker aims to steal the intellectual property of the model's owner.

Let us consider a binary classification problem where the LR model is trained on a dataset $\{(x_1y_1), (x_2y_2), \ldots, (x_ny_n)\}$, where $x_i$ is the input vector and $y_i$ is the true label. The LR model can be defined as follows:

$$p(Y = 1|X, \omega) = \sigma\left(\omega^T X + b\right),$$

where, $\sigma(z) = 1/(1 + \exp(-z))$ is the sigmoid function and $w$ is the weight vector learned during training.

Let us assume that the attacker has access to the trained model and a set of input–output pairs $\left\{\left(x'_1y'_1\right), \left(x'_2y'_2\right), \ldots, \left(x'_my'_m\right)\right\}$ that are generated by querying the model with inputs that the attacker chooses. The attacker's goal is to use this set of input–output pairs to reconstruct a copy of the trained model.

The attacker can use a technique called "model inversion" to extract the model's weight vector $\omega$. Specifically, the attacker can solve the following optimization problem:

$$\textit{minimize} \left\|\omega - \omega'\right\| \text{ subject to } p\left(y = 1|x', \omega'\right) = y',$$

where, $\omega'$ is the extracted weight vector, $x'$ is an input vector chosen by the attacker, and $y'$ is the output label predicted by the model for input $x'$. The objective is to find a weight vector $\omega'$ that produces the same output label as the original model for the chosen input $x'$.

This optimization problem can be solved using various techniques, such as gradient descent or genetic algorithms. By iteratively updating the weight vector $\omega'$, the attacker can gradually improve the extraction quality. Once the attacker has extracted the weight vector, they can reconstruct and use the LR model for their own purposes.

### 5.5.1 | HE-based solution to defend against model extraction attacks

To prove that HE can defend against extraction attacks on LR, we need to show that an attacker cannot extract the model parameters from the encrypted output predictions even if they can access the encrypted model and the plaintext input data.

Assuming that an attacker has access to the encrypted model parameters and the plaintext input data, they can use the encrypted model to make predictions on the input data and obtain the encrypted output predictions. The attacker's goal is to extract the model parameters from the encrypted output predictions without access to the decryption key.

Let us assume that the attacker has access to the encrypted model parameters $[\omega]$ and $[b]$, and the plaintext input data $X$. We can write the LR model as follows:

$$[Y] = \sigma\left(\left[\omega^T\right][X] + [b]\right).$$

Assuming that the encryption function Encryption() is a homomorphism, we can apply the homomorphic property of the encryption scheme to obtain the following:

$$[Y] = \left[\sigma\left(\omega^T X + b\right)\right].$$

Here, $\omega$ and $b$ represent the original weight vector and bias term, respectively.

Since the attacker cannot access the decryption key, they cannot directly obtain the original weight vector and bias term from the encrypted model parameters. However, the attacker can use the plaintext input data and the encrypted output predictions to perform an optimization-based attack to extract the model parameters.

The attacker's goal is to find the model parameters $\omega$ and $b$ that minimize the difference between the actual output predictions Enc($y$) and the predicted output predictions based on the extracted model parameters:

- Minimize $\left\|[Y] - \left[\sigma\left(\omega^T X + b\right)\right]\right\|_2$.
- Here, $\|.\|_2$ denotes the $L_2$ norm.

Assuming that the encryption function Encryption () is semantically secure, the attacker cannot obtain any useful information about the original input data or the model parameters from the encrypted output predictions. Therefore, the attacker cannot perform an optimization-based attack to extract the model parameters without access to the decryption key.

This proves that HE can defend against extraction attacks on LR, even if the attacker has access to the encrypted model parameters and the plaintext input data.

## 6 | EXPERIMENTS AND RESULT ANALYSIS

In this section, we present parameter sets with experimental results. Our implementation is based on the TenSEAL library[34] an open-source library to build CKKS encryption and decryption schemes for Privacy-Preserving Machine Learning using HE. All experiments were performed on an Intel-i5-10 500 CPU at a 3.10 GHz processor with four cores and 16 GB of RAM.

### 6.1 | Datasets

We experimented using banks from German, Taiwan, Japan, and Australia financial datasets[35–38] from UCI Machine Learning Repository. The datasets under consideration contain a solitary binary outcome variable, making them suitable for training binary classifiers such as LR. Table 2 provides an overview of the datasets, including the number of observations (rows) and features (columns). During the study, four subsets were employed for learning using a learning rate $\alpha$ of approximately 1, while one subset was reserved for testing the trained model.

**TABLE 2** Description of datasets.

| Dataset | Number of observations (rows) | Number of features (columns) |
|---|---|---|
| German | 1000 | 20 |
| Taiwan | 30 000 | 25 |
| Japan | 691 | 16 |
| Australia | 690 | 14 |

## 6.2 | Description of datasets

See Table 2.

## 6.3 | Parameters and timings for the homomorphic encryption scheme

Privacy-preserving Logistic Regression (PPLR) training via the HE technique faces difficulty in calculating the sigmoid function directly in the LR model. So, we adopted the least square polynomial approximation when evaluating the gradient descent algorithm. We adopted the degree 5 and 3 polynomial approximation $g(x)$ by which Kim et al.[33] used the least square approach to approximate the sigmoid function over the domain $[-5, 5]$. For Security settings, we used the polynomial degree 8192 with coefficient module bit sizes [40, 21, 21, 21, 21, 21, 21, 40] taken.

In Table 3, we evaluated our model performance based on the average running time of Encryption evaluation in each fold. For encrypting training, the dataset takes 14, 252, 10, and 8 for German, Taiwan, Japan, and Australia datasets. The encryption time depends on the number of observations, features, and data type. For encryption, training time takes 89, 1428, 70, and 57 per each epoch for degree 3 and 92, 1540, 78, and 59 s for degree 5. The time increases depending on the number of observations, features, type of data, and approximation degree.

Table 4 compares the models generated using our encrypted approach and unencrypted LR for the German, Taiwan, Japan, and Australia datasets. In the unencrypted cases, we employed the original sigmoid function on the same training dataset as in the encrypted cases. To evaluate the effectiveness of the models, we computed the accuracy (%) metric, defined as the percentage of correct predictions on the testing dataset, and Area Under the Curve (AUC) values by varying the epochs up to 20 for degree 3 and degree 5.

**TABLE 3** Dataset encryption time and encrypted training time.

| Dataset | Degree of $g(x)$ | Encryption of dataset (s) | Encryption training time per epoch (s) |
|---|---|---|---|
| German | 3 | 14 | 89 |
| | 5 | 14 | 92 |
| Taiwan | 3 | 252 | 1428 |
| | 5 | 252 | 1540 |
| Japan | 3 | 10 | 70 |
| | 5 | 10 | 78 |
| Australia | 3 | 8 | 57 |
| | 5 | 8 | 59 |

**TABLE 4** Comparison of unencrypted logistic regression and encrypted logistic regression.

| Dataset | Unencrypted logistic regression | | Encrypted logistic regression | | |
|---|---|---|---|---|---|
| | Accuracy (%) | AUC | Degree of $g(x)$ | Accuracy (%) | AUC |
| German | 80 | 0.73 | 3 | 75.4 | 0.7 |
| | 80 | 0.73 | 5 | 79.5 | 0.72 |
| Taiwan | 62 | 0.65 | 3 | 56.4 | 0.60 |
| | 62 | 0.65 | 5 | 60.4 | 0.63 |
| Japan | 81 | 0.83 | 3 | 78.2 | 0.8 |
| | 81 | 0.83 | 5 | 79.7 | 0.81 |
| Australia | 80 | 0.81 | 3 | 72.2 | 0.76 |
| | 80 | 0.81 | 5 | 78.5 | 0.78 |

The comparative analysis of the accuracy and AUC of Unencrypted Logistic Regression (UE_LR) and Logistic Regression using Homomorphic Encryption (LR_HE) for German, Taiwan, Japan, and Australia datasets are shown in Figure 4. The analysis was performed by using degree 3 and degree 5, and the experiment was conducted multiple times, with the average of those considered. The epochs were varied up to 20.

The results show that HE_LR_g3 has a deviation from UE_LR of around 3.6%, 5.6%, 1.8%, and 7.8%, while HE_LR_g5 has a trivial deviation from UE_LR of around 0.5%, 1.6%, 1.3%, and 1.5%. The AUC of HE_LR_g3 has a deviation from UE_LR of around 0.03–0.05, while HE_LR_g5 has a trivial deviation from UE_LR of around 0.01–0.03. In some instances, HE_LR_g5 outperforms UE_LR in both accuracy and AUC.

Table 5 presents the comparison of the proposed model with the existing models regarding data privacy such as training privacy and model privacy in training phase, and input and output privacy in inference phase.

It is evident from Table 5 that the proposed methodology provides privacy for various types of data in all phases.

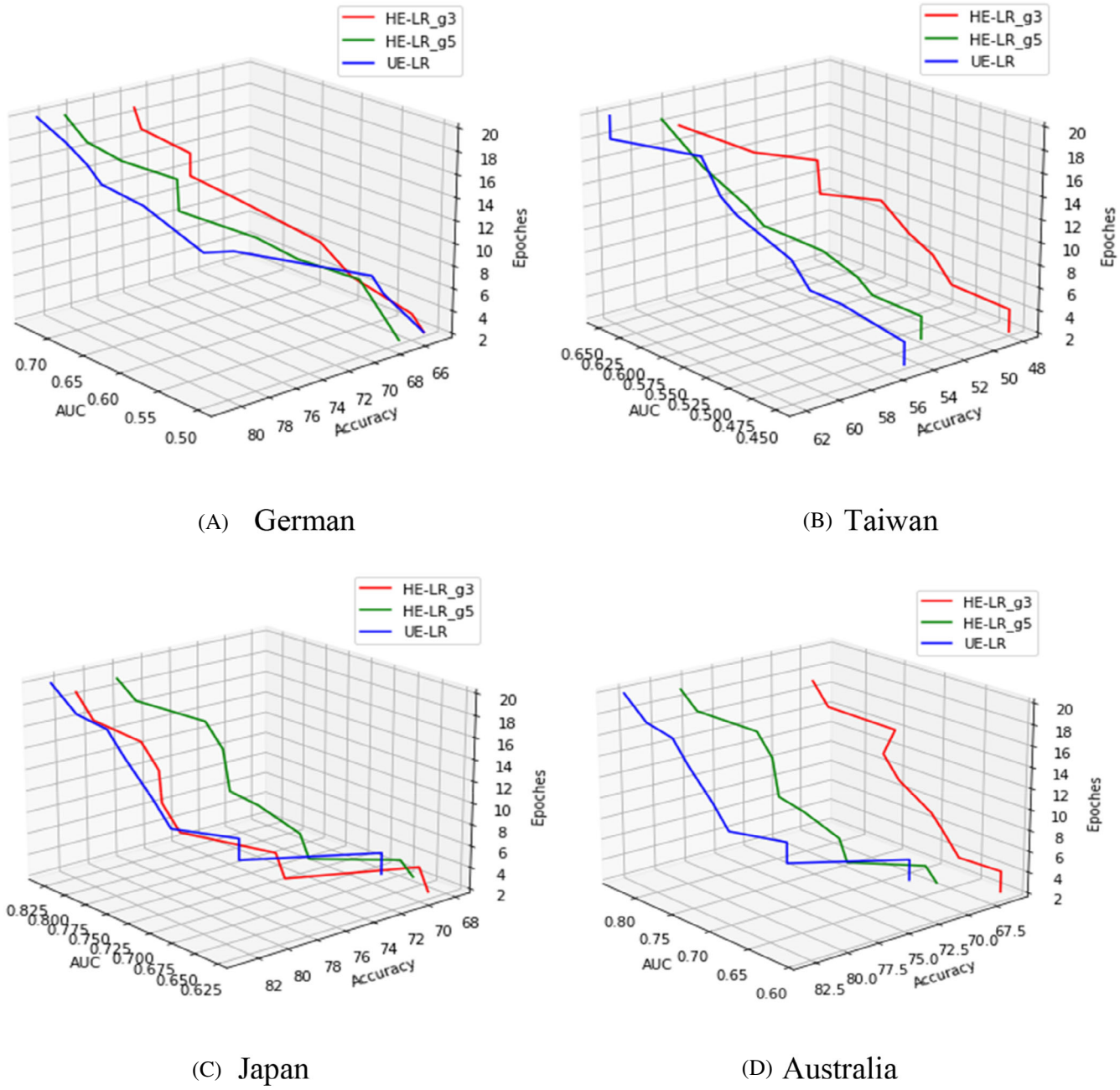


**FIGURE 4** Average accuracy and AUC of unencrypted logistic regression and encrypted logistic regression.

**TABLE 5** Comparative analysis.

| Type of data privacy | Li et al.[29] | Zheng et al.[27] | Khan et al.[28] | Chiang et al.[30] | Proposed |
|---|---|---|---|---|---|
| Training privacy | ✓ | × | × | ✓ | ✓ |
| Input privacy | ✓ | ✓ | ✓ | ✓ | ✓ |
| Model privacy | ✓ | ✓ | × | × | ✓ |
| Output privacy | × | × | ✓ | × | ✓ |

Our implementation revealed that using the degree 5 least squares polynomial for the gradient descent LR algorithm results in higher accuracy and AUC than using degree 3. This approach provides results comparable to unencrypted LR using the original sigmoid function with the same number of iterations.

## 7 | CONCLUSION AND FUTURE DIRECTIONS

In this article, we proposed a HE-aware logistic regression model built on encrypted data using CKKS. Our method can be used to outsource the training phase of LR to a cloud service in a privacy-preserving manner. We conducted experiments using the TenSEAL library on real UCI Machine Learning Repository datasets to evaluate the performance of both LR (over unencrypted data) and the proposed system. We made a security analysis that the proposed system can defend against poison, evasion, member inference, model inversion, and model extraction in respective stages of Machine learning. The average performance of the unencrypted LR for the German, Taiwan, Japan, and Australia is 80%, 62%, 81%, and 80%. The average performance of the proposed encrypted LR using HE approach with degree 3 for the German, Taiwan, Japan, and Australia is 75%, 56%, 78%, and 72%. The average performance of the proposed encrypted LR using HE approach with degree 5 for the German, Taiwan, Japan, and Australia is 79%, 60%, 79%, and 78%. The performance of the proposed approach provides satisfactory results relative to the model without HE, with a minimal difference in accuracy and AUC between encrypted and unencrypted methods. The HELR_g5 is highly adaptable for privacy-preserving credit risk analysis.

In the context of future advancements, the suggested system has the potential for expansion beyond a single financial institution to encompass multiple institutions. As the number of participants or data sources grows, there is a likelihood of enhanced model performance. This enhancement can be realized through the incorporation of a federated learning framework. The proposed system currently operates within a semi-trusted setup, with a focus on safeguarding user privacy from external computational interventions. In future iterations, there is an opportunity to evolve the system into a fully trusted environment by implementing a multikey framework.

### CONFLICT OF INTEREST STATEMENT

The authors declare that they have no competing interests.

### DATA AVAILABILITY STATEMENT

Data will be provided based on a request to the corresponding author.

### RESEARCH INVOLVING HUMAN PARTICIPANTS AND/OR ANIMALS

This article does not contain any studies with animals performed by any of the authors.

### ORCID

*Vankamamidi S. Naresh* https://orcid.org/0000-0002-6273-9041

### REFERENCES

1. Zhou M. Credit risk assessment modeling method based on fuzzy integral and SVM. *Mob Inform Syst*. 2022;2022:3950210. doi:10.1155/2022/3950210
2. Papouskova M, Hajek P. Two-stage consumer credit risk modelling using heterogeneous ensemble learning. *Decis Supp Syst*. 2019;118:33-45. doi:10.1016/j.dss.2019.01.002

3. Bussmann N, Giudici P, Marinelli D, Papenbrock J. Explainable machine learning in credit risk management. *Comput Econ*. 2021;57:203-216. doi:10.1007/s10614-020-10042-0
4. Moradi S, Mokhatab Rafiei F. A dynamic credit risk assessment model with data mining techniques: evidence from Iranian banks. *Financ Innov*. 2019;5:15. doi:10.1186/s40854-019-0121-9
5. Liu L. A self-learning bp neural network assessment algorithm for credit risk of commercial bank. *Wirel Commun Mob Comput*. 2022;2022:1-10. doi:10.1155/2022/9650934
6. Liu J, Zhang S, Fan H. A two-stage hybrid credit risk prediction model based on XGBoost and graph-based deep neural network. *Exp Syst Appl*. 2022;195:116624. doi:10.1016/j.eswa.2022.116624
7. Abdulsalam YS, Hedabou M. Security and privacy in cloud computing: technical review. *Fut Internet*. 2022;14(1):11. doi:10.3390/fi14010011
8. Balaji K, Manikandasaran SS. Data security and deduplication framework for securing and deduplicating users' data in public and private cloud environment. *J Sci Res*. 2022;14(1):153-165. doi:10.3329/jsr.v14i1.54063
9. Kumar S, Srivastava PK, Pal AK, et al. Protecting location privacy in cloud services. *J Discr Math Sci Cryptogr*. 2022;25(4):1053-1062. doi:10.1080/09720529.2022.2072430
10. Wang T, Xu L, Zhang M, Zhang H, Zhang G. A new privacy protection approach based on K-anonymity for location-based cloud services. *J Circ Syst Comput*. 2022;31(5):2250083. doi:10.1142/S0218126622500839
11. Ahmed A, Kumar S, Shah AA, Bhutto A. Cloud computing security issues and challenges. *Trop Sci J*. 2023;2(1):1-8.
12. Devi Satya Sri V, Vemuru S. A hybrid multi-user based data replication and access control mechanism for cloud data security. *SMART Technologies in Data Science and Communication: Proceedings of SMART-DSC 2022*. Springer Nature Singapore; 2023:91-100.
13. Darwish SM, Essa RM, Osman MA, Ismail AA. Privacy preserving data mining framework for negative association rules: an application to healthcare informatics. *IEEE Access*. 2022;10:76268-76280. doi:10.1109/ACCESS.2022.3192447
14. Jain P, Shakya D. *A Review of Different Privacy Preserving Techniques in Data Mining*; 2022. doi:10.2139/ssrn.4021149
15. Kenthapadi K, Mironov I, Thakurta AG. Privacy-preserving data mining in industry. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM; 2019:840-841. doi:10.1145/3289600.3291384
16. Kreso I, Kapo A, Turulja L. Data mining privacy preserving: research agenda. *Wiley Interdiscip Rev Data Min Knowl Discov*. 2021;11(1):e1392. doi:10.1002/widm.1392
17. Hesamifard E, Takabi H, Ghasemi M, Wright RN. Privacy-preserving machine learning as a service. *Proc Priv Enhanc Technol*. 2018;2018(3):123-142.
18. Abramson W, Hall AJ, Papadopoulos P, Pitropakis N, Buchanan WJ. A distributed trust framework for privacy-preserving machine learning. *International Conference on Trust and Privacy in Digital Business*. Springer, Cham; 2020:205-220.
19. Patra A, Suresh A. BLAZE: blazing fast privacy-preserving machine learning. *arXiv preprint arXiv:2005.09042*; 2020. doi:10.14722/ndss.2020.24202
20. Mamta BB, Gupta K-C, Li VCM, Leung KEP, Yamaguchi S. Blockchain-assisted secure fine-grained searchable encryption for a cloud-based healthcare cyber-physical system. *IEEE/CAA J Autom Sin*. 2021;8(12):1877-1890. doi:10.1109/JAS.2021.1004003
21. Al Issa HA, Al-Jarah MH, Almomani A, Al-Nawasrah A. Encryption and decryption cloud computing data based on XOR and genetic algorithm. *Int J Cloud Appl Comput (IJCAC)*. 2022;12(1):1-10.
22. Devi KJ, Singh P, Thakkar HK, Kumar N. Robust and secured watermarking using ja-fi optimization for digital image transmission in social media. *Appl Soft Comput*. 2022;131:109781.
23. Aono Y, Hayashi T, Phong LT, Wang L. Privacy-preserving logistic regression with distributed data sources via homomorphic encryption. *IEICE Trans Inform Syst*. 2016;99(8):2079-2089. doi:10.1587/transinf.2015INP0020
24. Kim M, Song Y, Wang S, Xia Y, Jiang X. Secure logistic regression based on homomorphic encryption. *Cryptology ePrint Archive, Report 2018/074*; 2018. doi:10.2196/medinform.8805 https://eprint.iacr.org/2018/074
25. Mohassel P, Zhang Y. Secureml: a system for scalable privacy-preserving machine learning. *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22–26, 2017*. IEEE Computer Society; 2017:19-38. doi:10.1109/SP.2017.12
26. Wang S, Zhang Y, Dai W, et al. Healer: homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. *Bioinformatics*. 2016;32(2):211-218. doi:10.1093/bioinformatics/btv563
27. Zheng Y, Wu Z, Yuan Y, Chen T, Wang Z. PCAL: a privacy-preserving intelligent credit risk modeling framework based on adversarial learning. *arXiv preprint arXiv:2010.02529*; 2020.doi:10.48550/arXiv.2010.02529
28. Khan T, Bakas A, Michalas A. Blind faith: privacy-preserving machine learning using function approximation. *2021 IEEE Symposium on Computers and Communications (ISCC)*. IEEE; 2021:1-7.
29. Li J, Kuang X, Lin S, Ma X, Tang Y. Privacy preservation for machine learning training and classification based on homomorphic encryption schemes. *Inform Sci*. 2020;526:166-179.
30. Chiang J. Privacy-preserving logistic regression training with a faster gradient variant. *arXiv preprint arXiv:2201.10838*; 2022.
31. Gentry C. Fully homomorphic encryption using ideal lattices. *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. ACM; 2009:169-178.
32. Cheon JH, Kim A, Kim M, Song Y. Homomorphic encryption for arithmetic of approximate numbers. *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3–7, 2017, Proceedings, Part I 23*. Springer International Publishing; 2017:409-437.
33. Kim M, Song Y, Wang S, Xia Y, Jiang X. Secure logistic regression based on homomorphic encryption: design and evaluation. *JMIR Med Inform*. 2018;6(2):e8805. doi:10.2196/medinform.8805

34. Benaissa A, Retiat B, Cebere B, Belfedhal AE. Tenseal: a library for encrypted tensor operations using homomorphic encryption. *arXiv preprint arXiv:2104.03152*; 2021. doi:10.48550/arXiv.2104.03152
35. Merz CJ, Murphy P. *UCI Repository of Machine Learning Databases*. https://archive.ics.uci.edu/ml/datasets/
36. Lichman M. *UCI Machine Learning Repository*. University of California, School of Information and Computer Science; 2013. http://archive.ics.uci.edu/ml
37. http://archive.ics.uci.edu/ml/datasets/Japanese+Credit+Screening
38. http://archive.ics.uci.edu/ml/datasets/Statlog+%28Australian+Credit+Approval

**How to cite this article:** Divakar Allavarpu VVL, Naresh VS, Krishna Mohan A. Privacy-preserving credit risk analysis based on homomorphic encryption aware logistic regression in the cloud. *Security and Privacy*. 2024;7(3):e372. doi: 10.1002/spy2.372