Peer-to-Peer Networking and Applications (2025) 18:138
https://doi.org/10.1007/s12083-025-01963-4

Enabling privacy-preserving and distributed intelligent credit scoring by zero-knowledge proof and functional encryption

Yangyang Bao[1,5] · Lingrui Pan[2] · Xiaochun Cheng[3] · Liming Nie[4]

Received: 20 December 2024 / Accepted: 23 March 2025 / Published online: 2 April 2025

Abstract
Credit scores, as quantifiable indicators of individual creditworthiness reflected in credit reports, are widely adopted by financial institutions as prerequisites for accessing various financial services. Therefore, the accuracy and objectivity of credit scores are paramount in the credit reporting system. To achieve this, there is a need to optimize scoring models while broadening the sources of credit data. However, the circulation of distributed credit data may expose sensitive customer information. Existing solutions, which protect privacy through secure multi-party computation and homomorphic encryption, introduce substantial computational costs and face challenges in resolving trust issues among multiple parties dealing with encrypted data. In response, this paper introduces a privacy-preserving credit scoring system. Initially, two range proof schemes for weights and credit data, as well as a proof scheme for the association between scores and ciphertexts, are devised to address trust concerns among parties. Building on these foundations, a privacy-preserving credit scoring (PPCS) scheme tailored for traditional models and a privacy-preserving intelligent credit scoring (PICS) scheme are constructed using inner-product functional encryption (IPFE). These schemes notably enhance the efficiency of encrypted scoring computations and encrypted vector operations in neural network model training, while introducing a novel approach to credit score prediction based on neural networks. This paper also undertakes rigorous security analyses of the proposed schemes. Experimental and performance evaluations demonstrate that PPCS and PICS exhibit superior computational and storage efficiency compared to related schemes, with PICS demonstrating exceptional performance in both binary classification prediction of default risk and credit score prediction.

Keywords Distributed credit scoring · Functional encryption · Neural networks · Privacy protection · Zero-knowledge proof

1 Introduction

Credit system entails the collection and organization of credit information on individuals or enterprises by specialized credit evaluation agencies, culminating in the production of credit report. These reports are subsequently utilized by financial institutions and various societal sectors to assess credit risk and make transactional decisions. In the United States, credit agencies such as Experian, Equifax, and TransUnion hold pivotal positions [1]; whereas in Europe, there is a stronger reliance on credit systems established by central banks. These systems not only enhance the transparency and efficiency of credit markets but also aid financial institutions in risk management, improving returns, and promoting the healthy development of the credit economy.

Extended author information available on the last page of the article

Credit scoring [2], as a key component of credit report, visually and quantitatively presents customers' credit standing to financial institutions without disclosing their personal information, borrowing history, or consumption records. Financial institutions frequently employ credit scoring as an eligibility criterion for loan or leasing services. Hence, accurately calculating credit scores is an indispensable aspect of achieving financial equity [3]. A typical distributed credit scoring architecture is abstracted and illustrated in Fig. 1. Within this architecture, distributed credit data providers dispatch credit data $(x_1, ..., x_{12})$ in response to requests from the credit bureau. Upon receiving credit query from the creditor, the credit bureau calculates the credit score $score = \sum_{i=1}^{t} w_i x_i$ based on the credit data $(x_1, ..., x_{12})$ and universally applied scoring weights $(w_1, ..., w_{12})$, and provide these credit scoring service to the creditor. The creditor then grants loans to the debtors based on their scores.

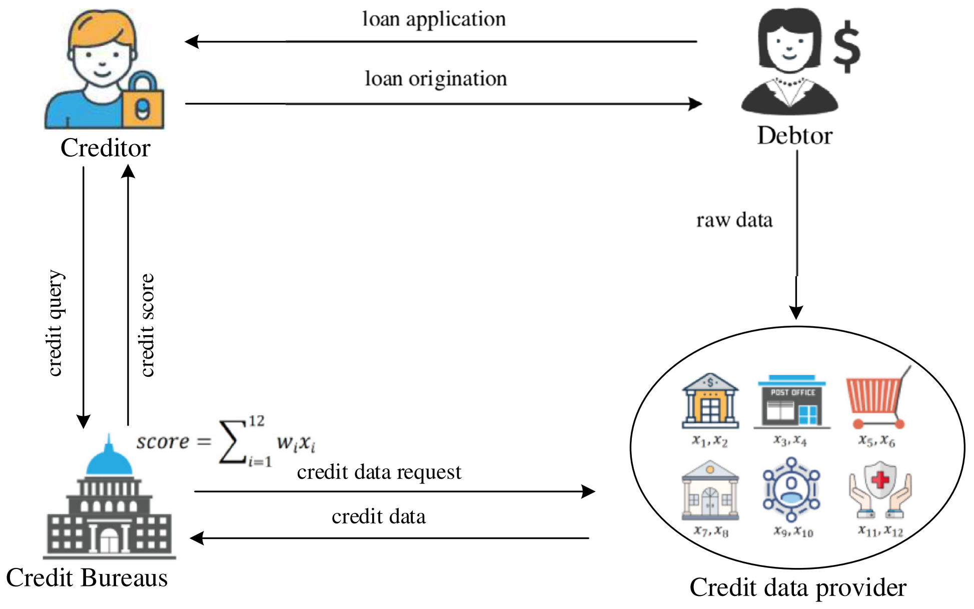


Fig. 1 Architecture of a distributed credit system

To achieve more precise, scientific, and equitable credit scoring, improvements must be made on two fronts: optimizing the scoring model and expanding data sources. Regarding the former, besides optimizing weights through application verification, some scholars in recent years have also explored credit evaluation based on machine learning [4], such as risk prediction based on logistic regression and random forests. As for the latter, it necessitates enhanced data sharing, which entails incorporating more alternative data sources into credit scoring calculations beyond the credit history reported by financial institutions. These alternative data sources include e-commerce transactions, payment history, medical records, social security information, social media activities, and judicial records.

However, the aforementioned approach raises concerns regarding security, privacy, trust and efficiency [5], specifically as follows:

- *Security and privacy issues of data and weights*: The extensive collection and uncontrolled use of credit data may lead to the disclosure of customer privacy, thereby jeopardizing customers' enjoyment of equal rights and even causing economic or reputational losses to them. For instance, customers may face employment discrimination or illegal tracking and surveillance due to the leakage of their personal information, while leaked financial data might be sold to business competitors, resulting in the disclosure of trade secrets. In response, privacy protection regulations, including the General Data Protection Regulation (GDPR) [6], impose strict constraints on data security and user privacy in financial activities, covering aspects such as data collection, storage, usage, and destruction. Credit data providers are also reluctant to provide data due to concerns about becoming involved in legal disputes related to user privacy. Similarly, the disclosure of scoring weights employed by credit bureaus may attract special attention from attackers towards certain data items, and even criminals might reverse-engineer credit data by combining scores and weights. Furthermore, the weights also represent the industry experience and intellectual property of credit bureaus, prompting them to consistently keeps the weights confidential.
- *Trust issues*: To address the above security and privacy issues in this scenario, a promising solution is resorting to secure multi-party computation. It allows for the collaborative computation of credit scores without any party (credit data providers, credit bureaus, and creditors) revealing their sensitive information, namely credit data and weights. However, the establishment of trust among these parties in the absence of data transparency remains a challenge. Lin et al. [26] have developed a secure credit scoring system based on homomorphic encryption [7]. This system encrypts weights and credit data separately, then computes the sum of their ciphertexts, and leverages the property of additive homomorphism to retrieve the final score. Additionally, this scheme provides such "trust" by incorporating non-interactive zero-knowledge (NIZK) proofs to enable credit data providers to verify weights, credit bureaus to authenticate data, and creditors to validate the correlation between scores and ciphertexts. However, the heavy computational overhead associated with homomorphic encryption poses a significant obstacle to the efficiency of credit scoring computations.
- *Accuracy issues of credit scores*: The weights used in traditional credit scoring models, which are based on the operational experience of credit bureaus, may compro-

mise the accuracy and objectivity of credit scores. As we mentioned earlier, deviation in credit scoring can exclude individuals with repayment ability or those without historical credit records but intending to repay on time (known as "credit invisibles" [9]), thereby hindering financial equity.

• *Efficiency issues*: Existing few privacy-preserving credit scoring schemes including Lin et al.'s scheme [26] rely on additive homomorphic encryption (such as Paillier). However, homomorphic encryption schemes incur significant computational overhead, which is obviously impractical in a scoring task ivolves large-scale credit data.

Inspired by Lin et al.'s [26] work and neural networks [8], this paper proposes a privacy-preserving credit scoring system, which encompasses two independent schemes tailored specifically for privacy-preserving traditional credit scoring and privacy-preserving intelligent credit scoring scenarios. The former is based on a polynomial-based scoring model, while the latter utilizes a neural network for credit score prediction. The specific contributions of this paper are enumerated as follows:

• We have constructed fundamental proof schemes for weight verification, credit data verification, and the relationship between ciphertext and scores under functional encryption, based on zero-knowledge range proofs [21] and the NIZK protocol with Fiat-Shamir heuristic transformation [22], respectively. Specifically, range proofs for weights and credit data are employed to ensure that they fall within reasonable bounds, thereby preventing credit bureaus or credit data providers from extracting sensitive information by using or providing deceptive weights and credit data.

• We propose a Privacy-Preserving Credit Scoring (PPCS) scheme and a Privacy-Preserving Intelligent Credit Scoring (PICS) scheme. Specifically, PPCS leverages Inner-product functional encryption (IPFE) protocol [11] to convert traditional credit scoring into the inner product between encrypted credit data vectors and local weight vectors. Additionally, it employs three proposed basic proof schemes to address trust issues among parties involved with encrypted data and weights. On the other hand, PICS utilizes IPFE to support vector operations within activation functions during neural network training based on encrypted data, while only requiring a range proof for the initial weights of the neural network model. Beyond binary classification predictions of whether default occurs, to our knowledge, PICS represents the first scheme that supports credit score prediction.

• We have conducted a thorough security analysis and extensive experimental evaluations of PPCS and PICS. Results show that both schemes outperform the state-of-the-art in storage and computation costs. Leveraging IPFE, PPCS and PICS are highly efficient in batch credit scoring. PICS excels in binary classification prediction of default risk, with accuracy near 94% and AUC nearing 0.85 for ROC curves, and also demonstrates strong performance in credit score prediction.

The remainder of this paper is organized as follows: Section 2 presents an overview of related work, Section 3 introduces preliminaries, Section 4 shows three fundamental proof schemes, Section 5 describes the construction of two credit scoring schemes, Section 6 conducts a security analysis, Section 7 presents experimental analysis and performance evaluation, and Section 8 summarizes the entire paper.

2 Related works

Privacy-preserving credit scoring In recent years, the advancement of privacy computing and blockchain technology has greatly promoted the related research of privacy-preserving credit scoring. Qiao et al. [29] have designed a privacy-preserving credit evaluation system based on blockchain and the Phillie homomorphic encryption protocol, which facilitates the secure sharing of credit data, ensuring authenticity and traceability throughout the process. Addressing the linear characteristics of credit scoring, they further employ linear encryption processing of the raw data to reduce computational and storage overhead. Qiao et al. [30] also proposed a decentralized privacy-preserving credit evaluation system based on Hyperledger Fabric blockchain [31], which features trustworthy data content and computation. This system is divided into three primary functional components: identity management, off-chain encrypted data upload, and on-chain data secure sharing and aggregation. Specifically, by incorporating a ciphertext-policy attribute-based encryption (CP-ABE) [32] access control scheme, unauthorized access is prevented. Furthermore, the system ensures the security of multi-party data sharing and aggregation through on-chain data secure sharing and aggregation based on Paillier homomorphic encryption. In addition to traditional scoring, there are some AI-based credit scoring studies. Li et al. [33] constructed a credit evaluation framework based on federated learning, which permits financial institutions to engage in joint modeling without requiring the local data to be transferred, thereby bridging the "data islands". They also demonstrated that this framework exhibits satisfactory predictive performance across various machine learning models. However, the framework is limited to coarse binary classifications predicting the presence or absence of overdue risk and is not suitable for credit scoring. Similarly, He et al. [34] devised a secure multi-party scoring model grounded in a vertical federated logistic regression framework, employing homomorphic encryption to protect the security of the gradients. Jovanovic et al. [35] proposed an interpretable federated learning credit scoring framework based on a blockchain platform, addressing the issues of data island, data security, and scoring authority in credit scoring. However, this framework remains at the theoretical level, lacking evaluations of its security and practical performance. Lin et al. [26] developed a secure and trustworthy scoring mechanism based on zero-knowledge proofs and Paillier homomorphic encryption. However, the recognized performance issues of homomorphic encryption limit its practicality. Naresh [27] proposed a privacy-preserving binary risk credit prediction model based on deep neural networks, also utilizing Paillier homomorphic encryption to protect user privacy during the training process. In response to this limitation, Andolfo et al. [10] proposed a credit scoring scheme leveraging novel functional encryption and a trusted execution environment, which contributes to enhancing performance in high-concurrency credit scoring business scenarios.

Privacy-preserving machine learning with cryptographical protocols Privacy-preserving machine learning (PPML) [37] has garnered significant attention in the field of artificial intelligence in recent years, with cryptographic techniques representing the mainstream approach for its realization. Initially, PPML was primarily realized through fully homomorphic encryption [12] or secure multi-party computation (SMPC) [13]. With ingenious algorithmic constructions, lightweight semi-homomorphic or somewhat-homomorphic encryption algorithms, such as Paillier [14] and OU [15], have garnered increasing attention and discussion within the field of PPML. However, these schemes are still severely constrained by the computational overheads of encryption and decryption when training large-scale machine learning models. Xu et al. [16] have designed a privacy-preserving federated learning scheme that employs the stochastic gradient descent strategy for model updates. Gradient parameters are securely transmitted under the protection of homomorphic encryption. Additionally, Shamir threshold secret sharing scheme [17] is utilized to address the common issue of user dropout in federated learning. Xu et al. [18] implemented a model named CryptoNN, which utilizes functional encryption instead of homomorphic encryption for matrix operations in convolutional neural networks (CNN), thereby enhancing the training efficiency of encrypted CNN by over 20 times. Based on this, Panzade and Takabi [19] proposed an extended scheme named FENet, which employs the function hiding inner product encryption protocol to conceal private information within vectors, thereby further enhancing the privacy protection effectiveness. Ryffel et al. [20] proposed a PPML scheme based on quadratic function encryption. This scheme initially introduces a quadratic function that enables data owners to perform computations without revealing the underlying data content.

3 Preliminaries

3.1 System model and threat model

We propose a Privacy-Preserving Credit Scoring (PPCS) scheme and a Privacy-Preserving Intelligent Credit Scoring (PICS) scheme. These schemes are constructed under the IPFE protocol, utilizing traditional scoring models and neural network-based scoring mechanisms respectively, to establish secure and privacy-preserving credit scoring systems. During this process, zero-knowledge proof schemes tailored for weights, encrypted credit data, and credit scores are constructed based on Zero-Knowledge Range Proof and NIZK proof protocols, respectively. These schemes address the trust issues between credit bureaus and financial institutions. The system models for these schemes are illustrated in Fig. 2.

The proposed system primarily involves interactions among three types of entities: the key management authority (KMA), the credit bureau, the financial institution and the creditor. Among them, the KMA serves as a fully trustworthy entity responsible for generating the system's public parameters, encompassing the common reference string (CRS), cryptographic public parameters, and the master secret key of the IPFE protocol. A financial institution serves as a trusted entity that possesses credit data and pertinent personal information of loan applicants, and submits such information to credit bureaus as required. All data are privacy-sensitive, as stipulated by the GDPR: Financial institutions must implement appropriate technical and organizational measures to protect the integrity and confidentiality of data, preventing unauthorized or unlawful processing, as well as accidental loss, destruction, or damage. The credit bureau, another trustworthy entity, is responsible for processing credit data and personal information to generate credit scores, which are then published as credit products for lenders to access. In collecting data, the credit bureau should adhere to GDPR's principle of "data minimization," meaning it should only gather the minimum amount of data necessary to fulfill its processing purposes, avoiding over-collection of user data. Similarly, it must rigorously protect the integrity and confidentiality of the data. Furthermore, data obtained from financial institutions can only be used for the purposes originally declared upon collection, unless new consent is obtained. A creditor, considered an untrusted entity, purchases scoring services from the credit bureau and decides whether to grant loans and credit limits to clients based on these scores. However, due to risk control needs, some creditors may attempt to obtain additional information beyond the scores, including customer account transaction histories and personal privacy details.

Figure 2-(a) illustrates the model of the proposed PPCS scheme. As depicted, KMA initially generates public param-

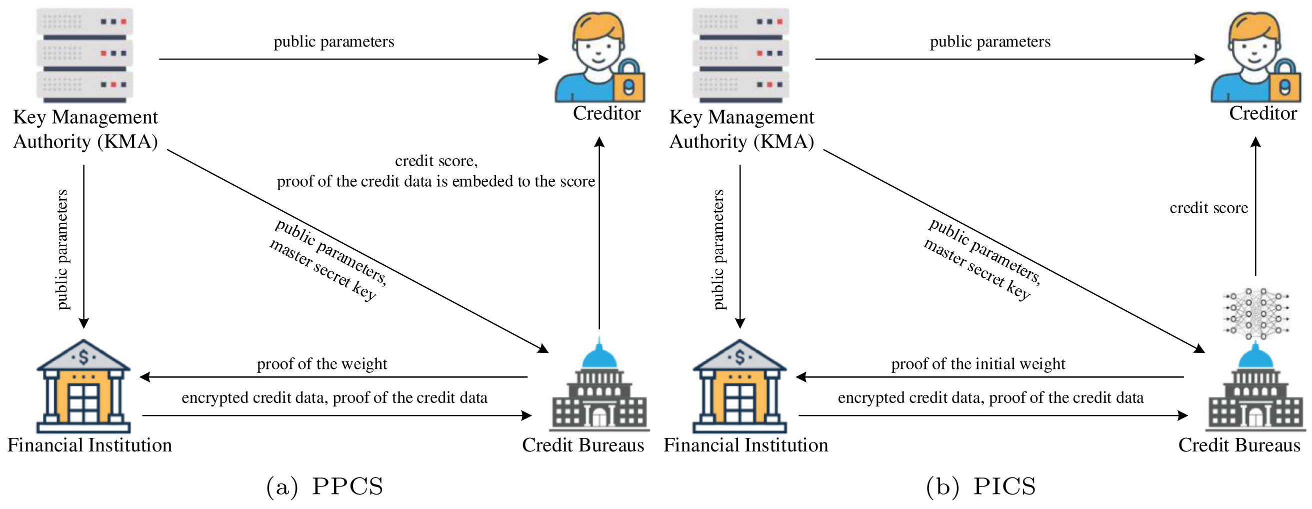

public parameters
Key Management Authority (KMA)
Creditor
credit score, proof of the credit data is embeded to the score
public parameters
public parameters, master secret key
proof of the weight
encrypted credit data, proof of the credit data
Financial Institution
Credit Bureaus
(a) PPCS
public parameters
Key Management Authority (KMA)
Creditor
credit score
public parameters
public parameters, master secret key
proof of the initial weight
encrypted credit data, proof of the credit data
Financial Institution
Credit Bureaus
(b) PICS

Fig. 2 System models

eters, which are then disclosed to all entities within the system. Additionally, the KMA generates a system master key and distributes it to the credit bureau. Subsequently, the credit bureau produces a range proof regarding the weights in its scoring model and sends this proof to the Financial Institution. The Financial Institution verifies the proof, encrypts the credit data, and sends it back to the credit bureau along with a range proof of the credit score. The credit bureau verifies this proof, locally decrypts the data to obtain the credit score, and then generates a proof that the credit data has been embedded with the score, which is subsequently provided to the creditor. The creditor verifies the proof and obtains the credit score. Figure 2-(b) depicts the model of the proposed PICS scheme. Following the same procedures as PPCS for public parameter and key distribution, initial weight proof generation, credit data encryption, and proof issuance, the credit data is then fed into a neural network model. Ultimately, the credit score is returned to the creditor.

3.2 Inner-product functional encryption

Inner Product Function Encryption (IPFE) was first proposed by Abdalla et al. [11] in 2015. Its core principle lies in preserving the ability to compute inner products of data while ensuring data confidentiality. Specifically, this technique encrypts plaintext data to generate corresponding ciphertext, and a decryptor uses a key corresponding to one vector to decrypt the ciphertext corresponding to another vector, thereby obtaining the inner product of the two vectors without revealing any other information about the vectors represented by the ciphertext. The specific algorithmic construction of IPFE is as follows:

- **IPFE.Setup** $(\lambda, l \rightarrow mpk, msk)$: Input the security parameter λ and the vector length l, it designates a multiplicative cyclic group $\mathbb{G}$ with prime order p, randomly selects g as a generator of $\mathbb{G}$. Then, it randomly chooses a vector $\mathsf{s} = (s_1, ..., s_l) \in \mathbb{Z}_p^l$, and calculates $h_i = g^{s_i}$, where $i \in [1, l]$. This algorithm outputs the public parameter $mpk = \{h_i\}_{i \in [1,l]}$ and the master secret key $msk = \mathsf{s}$.
- **IPFE.Encrypt** $(mpk, \mathsf{x} \rightarrow ct)$: Input a message vector $\mathsf{x} = (x_1, ..., x_l) \in \mathbb{Z}_p^l$, then it randomly chooses $r \in \mathbb{Z}_p$ and calculates $c_0 = g^r$, and also computes $c_i = h_i^r \cdot g^{x_i}$ for all $i \in \{1, l\}$. This algorithm outputs the ciphertext $ct = (c_0, \{c_i\}_{i \in \{1,l\}})$.
- **IPFE.KeyGen** $(msk, \mathsf{y} \rightarrow sk_f)$: Input a vector $\mathsf{y} = (y_1, ..., y_l) \in \mathbb{Z}_p^l$ and the master secret key msk, this algorithm outputs the decryption key $sk_f = (\mathsf{y}, \mathsf{s})$.
- **IPFE.Decrypt** $(sk_f, ct \rightarrow g^{\langle \mathsf{x}, \mathsf{y} \rangle})$: Input the ciphertext ct and the decryption key sk_f, this algorithm outputs the discrete logarithm in basis g: $\prod_{i \in [1,l]} c_i^{y_i} / c_0^{sk_f} = \prod_{i \in [1,l]} (g^{s_i r + x_i})^{y_i} / g^{r(\sum_{i \in [1,l]} y_i s_i)} = g^{\langle \mathsf{x}, \mathsf{y} \rangle}$.

3.3 Hardness assumption

Suppose $\mathbb{G}$ is a group of prime order q, and g is a generator of $\mathbb{G}$. For randomly chosen $x, y, z \in \mathbb{Z}_q$, where $\mathbb{Z}_q$ denotes the set of integers modulo q, consider the following two distributions:

The random quadruple $R = (g, g^x, g^y, g^{xy}) \in \mathbb{G}_4$;

The Decisional Diffie-Hellman (DDH) quadruple $R = (g, g^x, g^y, g^z) \in \mathbb{G}_4$.

If these two distributions are computationally indistinguishable, i.e., there does not exist a polynomial-time algorithm that can distinguish between them with non-negligible probability, then the DDH assumption is said to hold in group $\mathbb{G}$.

3.4 Camenisch's Zero-knowledge range proof

Camenisch et al. [21] provide a NIZK protocol for range proof by integrating classical NIZK proof scheme, Σ-protocol [23] and Boneh-Boyen signature scheme [24]. This protocol enables the prover to demonstrate that the secret value falls within a certain range without compromising confidentiality. To prove that the secret value σ lies within the interval $[0, u^l]$, where $u, l \in \mathbb{N}$, it first provides two randomly generator g and h of the group $\mathbb{G}$ along with a commitment C. The prover P should input the secret value σ and a random value r such that $C = g^{\sigma} h^{r}$, and runs the following interactive steps with the verifier V.

Step 1: V randomly selects $x \in \mathbb{Z}_p$ then calculates and sends $y = g^x$ and $A_i = g^{\frac{1}{x+i}}$ for each $i \in \mathbb{Z}_u$ to P.

Step 2: P randomly chooses $v_j \in \mathbb{Z}_p$ then calculates and sends $V_j = A_{\sigma_j}^{v_j}$ for each $j \in \mathbb{Z}_l$ such that $\sigma = \sum_j (\sigma_j u^j)$. Both the verifier and prover can compute the commitment $C = h^r \prod_j (g^{u^j})^{\sigma_j}$ and $V_j = g^{\frac{v_j}{x+\sigma_j}}$.

Step 3: P randomly chooses $s_j, t_j, m_j \in \mathbb{Z}_p$ for each $j \in \mathbb{Z}_l$ then calculates and sends $a_j = e(V_j, g)^{-s_j} e(g, g)^{t_j}$, $D = \prod_j (g^{u^j s_j}) h^{m_j}$.

Step 4: V randomly selects $c \in \mathbb{Z}_p$ and sends it to P.

Step 5: P calculates $z_{\sigma_j} = s_j - \sigma_j c$, $z_{v_j} = t_j - v_j c$ for each $j \in \mathbb{Z}_l$ and $z_r = m - rc$ and sends them to V. V then verifies whether there exists $D = C^c h^{z_r} \prod_j (g^{u^j z_{\sigma_j}})$ and $a_j = e(V_j, y)^c \cdot e(V_j, g)^{-z_{\sigma_j}} \cdot e(g, g)^{z_{v_j}}$ for each $j \in \mathbb{Z}_l$.

3.5 Neural networks

Neural network is a computational model that mimics the structure of neurons in the human brain, utilized for pattern recognition and processing of complex data. It represents one of the core technologies in the fields of machine learning and deep learning. The architecture of a neural network typically comprises multiple layers, including an input layer, hidden layers, and an output layer. Each layer contains multiple neurons, which are interconnected through weights and biases to facilitate information transmission and processing.

• Input Layer: The input layer serves as the starting point of a neural network, responsible for receiving external input data. The number of neurons in the input layer generally corresponds to the number of features in the input data.

• Hidden Layer: Positioned between the input and output layers, the hidden layer constitutes the core component of a neural network. The neurons within the hidden layer process the input data through nonlinear transformations and extract features. There can be multiple hidden layers, with each layer further abstracting and representing the input data.

• Output Layer: The output layer is the final layer of a neural network, tasked with outputting the processed results. The number of neurons in the output layer typically corresponds to the number of prediction targets.

In a neural network, each neuron receives input signals from other neurons and applies weights to these signals for summation. If the result of this weighted summation exceeds a certain threshold, the neuron becomes activated and generates an output signal, which is then passed to the neurons in the next layer. The connections between neurons are represented by weight matrices, which are parameters that the neural network learns during the training process. To introduce nonlinearity, activation functions are commonly added before the output of neurons. Common activation functions include ReLU, Sigmoid, and tanh, among others. These activation functions constrain the output of neurons within a certain range and introduce nonlinear characteristics, thereby enhancing the expressive power of the neural network. Below is a typical structure of an activation function:

$$a = f(\sum_{i=1}^{n} w_i^T \cdot x_i + b)$$

where w_i denotes the weight, x_i denotes the input of a neuron, and b is the bias.

Forward propagation and backward propagation are two fundamental processes in neural networks, playing crucial roles in both training and prediction phases of neural networks.

Forward propagation is the basic process in neural networks during both training and prediction. In this process, input data is passed through the layers of the network, where each layer performs certain computations and transformations on the data, ultimately producing an output result.

Backward propagation is the process used in neural networks for optimizing model parameters. Its core idea is to update the weights and biases of the model by computing the gradients of the loss function relative to the model parameters, thereby minimizing the loss function. Specifically, the process of backward propagation can be described as follows:

• Loss Calculation: The value of the loss function is computed based on the discrepancy between the predicted values and the true values. This value indicates the magnitude of the error of the model for the current input.

• Error Backpropagation: Using the chain rule, the derivatives of the error relative to the activation functions of each hidden layer are computed layer by layer. The key step here is to propagate the error backward from the output layer to compute the error for each neuron in the hidden layers.

• Gradient Computation: For each layer, the gradients of the error relative to the weights and biases are computed. This can be achieved by multiplying the error by the derivative of the activation function and the output of the previous layer.

• Parameter Update: The weights and biases of the network are updated based on the computed gradients and a learning rate (a predefined hyperparameter), causing the value of the loss function to decrease along the negative direction of the gradient, thereby optimizing model performance.

The backward propagation algorithm typically iterates over the entire training dataset multiple times within one training epoch. Each iteration updates the parameters until the loss function converges or a preset maximum number of iterations is reached.

4 Fundamental proof schemes

We first devise a proof protocol oriented towards traditional credit scoring models to address the trust issues between credit bureaus and financial institutions. The traditional credit scoring model can be abstracted as: $f(k_1, ..., k_t, m_1, ..., m_t) = \sum_{i=1}^{t}(k_i \cdot m_i)$, where $\{k_i\}_{i\in[1,t]}$ and $\{m_i\}_{i\in[1,t]}$ denote the weight and the credit data respectively. In order to integrate the aforementioned scoring model into Camenisch's Zero-Knowledge Range Proof scheme [21], modifications to the forms of weights and credit data are required: assume the range proof of weights and credit data as $k_i \in [1, u^l]$ and $m_i \in [1, u^l]$ for $i \in [1, t]$, then we have $k_i = \sum_{i=1}^{l-1}(k_{i,j} \cdot u^j)$, $m_i = \sum_{i=1}^{l-1}(m_{i,j} \cdot u^j)$, where $k_{i,j}, m_{i,j} \in [0, u]$. In the whole process from collecting credit data to return credit score, we use Camenisch's zero-knowledge range proof to construct the proof scheme for the weight (PSW), proof scheme for the credit data (PSCD) and proof scheme for the credit score (PSCS) respectively.

4.1 Proof scheme for the weight

This scheme enables the credit bureau to prove its weights statement $w_0 = (k_1, ..., k_t)$. The PSW scheme consists of three algorithms: **PSW-KeyGen**, **PSW-Proof** and **PSW-Verify**, they are constructed as follows.

- **PSW-KeyGen** $(\lambda) \rightarrow (pp, sk)$: This algorithm takes the security parameter λ as input, generates a CRS crs, then produces two multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ with prime order p, g denotes two generators of group $\mathbb{G}$. It also defines a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ and a collision-assistant hash function H. Then it invokes the Boneh-Boyen signature scheme [24] to obtain the public/private key pair $(pk = g^x, sk = x)$ and generates the signature $S_\tau = g^{\frac{1}{x+\tau}}$ for each element $\tau \in [0, u]$. This algorithm finally outputs the public parameters $pp = (p, \mathbb{G}, \mathbb{G}_T, g, e, pk, \{S_\tau\}, \mathsf{H})$ and the private key sk.
- **PSW-Proof** $(pp, w_0) \rightarrow (x_0, \pi_0)$: This algorithm takes the public parameters pp as input, then randomly selects $v_{i,j} \in \mathbb{Z}_p$ and calculates $V_{i,j} = S_{k_{i,j}}^{v_{i,j}} = g^{\frac{v_{i,j}}{x+k_{i,j}}}$. On this basis, the credit bureau generates two vectors $w_0 = (k_1, ..., k_t)$ and $x_0 = (c_1, ..., c_t)$, where $c_i = g^{\sum_{j=0}^{l-1}(k_{i,j}u^j)}$, then randomly selects $\alpha_{i,j}, \beta_{i,j}, \gamma_i \in \mathbb{Z}_p$ and calculates $a_{i,j} = e(V_{i,j}, g)^{-\alpha_{i,j}} \cdot e(g, g)^{\beta_{i,j}}$, $\chi_i = \prod_{j=0}^{l-1}(g^{\alpha_{i,j}u_j})$. It then generates $\eta_0 = \mathsf{H}(w_0||V_{1,0}||...||V_{t,l-1}||a_{1,0}||...||a_{t,l-1}||\chi_1||...||\chi_t)$, and $z_{k_{i,j}} = \alpha_{i,j} - k_{i,j}\eta_0$, $z_{v_{i,j}} = \beta_{i,j} - v_{i,j}\eta_0$ for $i \in [1, t]$ and $j \in [0, l]$. This algorithm finally assembles the proof $x_0 = (c_1, ..., c_t)$ and $\pi_0 = (\eta_0, \{V_{i,j}, a_{i,j}, \chi_i, z_{k_{i,j}}, z_{v_{i,j}}\}_{i\in[1,t], j\in[0,l-1]})$.
- **PSW-Verify** $(pp, x_0, \pi_0) \rightarrow (1/0)$: This algorithm takes the proof x_0, π_0 and the public parameters pp as input, then first retrieves the value of η_0 by computing $\eta_0' = \mathsf{H}(w_0||V_{1,0}||...||V_{t,l-1}||a_{1,0}||...||a_{t,l-1}||\chi_1||...||\chi_t)$. If $\eta_0' \neq \eta_0$, it directly returns 0 to indicate the proof is invalid. Otherwise, it checks whether the following equations hold: $\chi_i = c_i^{\eta_0'} \prod_{j\in[0,l-1]}(g^{u^j z_{k_{i,j}}})$ and $a_{i,j} = e(V_{i,j}, y)^{\eta_0'} \cdot e(V_{i,j}, g)^{-z_{k_{i,j}}} \cdot e(g, g)^{z_{v_{i,j}}}$ for $i \in [1, t]$ and $j \in [0, l-1]$, where $y = \prod_{i=1}^{t} g^{k_i}$. If they hold, it outputs 1 to indicate the proof is valid, otherwise outputs 0.

4.2 Proof scheme for the credit data

This scheme enables the credit bureau to prove its credit scores are computed with the correct credit data $w_1 = (m_1, ..., m_t)$. The PSW scheme consists of three algorithms: **PSCD-KeyGen**, **PSCD-Proof** and **PSCD-Verify**, they are constructed as follows.

- **PSCD-KeyGen** $(\lambda) \rightarrow (pp, sk)$: This algorithm is the same as the **PSW-KeyGen** algorithm, it additionally chooses a vector $\mathsf{s} = (s_1, ..., s_l) \in \mathbb{Z}_p^l$, and calculates $h_i = g^{s_i}$, where $i \in [1, l]$. Finally, this algorithm outputs the public parameters $pp = (p, \mathbb{G}, \mathbb{G}_T, g, e, pk, \{S_\tau\}_{i=1}^t, \{h_i\}_{i=1}^t, \mathsf{H})$ and the private key sk, s.
- **PSCD-Proof** $(pp, w_1) \rightarrow (x_1, \pi_1)$: This algorithm takes the public parameters pp as input, then randomly selects $v_{i,j}, r \in \mathbb{Z}_p$ and calculates $V_{i,j} = S_{m_{i,j}}^{v_{i,j}} = g^{\frac{v_{i,j}}{x+m_{i,j}}}$. On this basis, the credit bureau generates two vectors $w_1 = (m_1, ..., m_t)$ and $x_1 = (c_1, ..., c_t)$, where $c_i = h_i^r g^{\sum_{j=0}^{l-1}(m_{i,j}u^j)}$, then randomly selects $\alpha_{i,j}, \beta_{i,j} \in \mathbb{Z}_p$ and calculates $a_{i,j} = e(V_{i,j}, g)^{-\alpha_{i,j}} \cdot e(g, g)^{\beta_{i,j}}$, $\chi_i = \prod_{j=0}^{l-1}(g^{u^j\alpha_{i,j}})h_i^{\gamma_i}$. It then generates $\eta_1 = \mathsf{H}(w_1||V_{1,0}||...||V_{t,l-1}||a_{1,0}||...||a_{t,l-1}||\chi_1||...||\chi_t)$, and $z_{m_{i,j}} = \alpha_{i,j} - m_{i,j}\eta_1$, $z_{v_{i,j}} = \beta_{i,j} - v_{i,j}\eta_1$, $z_{h_i} = h_i^{r\eta_1+\gamma_i}$ for $i \in [1, t]$ and $j \in [0, l]$. This algorithm finally outputs the proof $x_1 = (c_1, ..., c_t)$ and $\pi_1 = (\eta_1, \{\chi_i, V_{i,j}, a_{i,j}, z_{m_{i,j}}, z_{v_{i,j}}, z_{h_i}\}_{i\in[1,t], j\in[0,l-1]})$.
- **PSCD-Verify** $(pp, x_1, \pi_1) \rightarrow (1/0)$: This algorithm takes the proof x_1, π_1 and the public parameters pp as input, then first retrieves the value of η_1 by computing $\eta_1' = \mathsf{H}(w_1||V_{1,0}||...||V_{t,l-1}||a_{1,0}||...||a_{t,l-1}||\chi_1||...||\chi_t)$. If $\eta_1' \neq \eta_1$, it directly returns 0 to indicate the proof is invalid. Otherwise, it checks whether the following equations hold: $\chi_i = c_i^{-\eta_1'} \cdot \prod_{j=0}^{l-1}(g^{u^j z_{m_{i,j}}}) \cdot z_{h_i}$ and $a_{i,j} = e(V_{i,j}, y)^{\eta_1'} \cdot e(V_{i,j}, g)^{-z_{m_{i,j}}} \cdot e(g, g)^{z_{v_{i,j}}}$ for $i \in [1, t]$ and $j \in [0, l-1]$, where $y = \prod_{i=1}^{t} g^{m_i}$. If they hold, it outputs 1 to indicate the proof is valid, otherwise output 0.

4.3 Proof scheme for the credit score

This scheme enables the credit bureau to demonstrate that the credit scores it outputs are related to the encrypted credit data. Since the encrypted credit data is not constrained by a specific range and the credit scores are directly visible to financial institutions or individual credit applicants, the aforementioned range proof scheme is not employed here. Instead, we construct the scheme based on the NIZK protocol with Fiat-Shamir heuristic transformation [25]. This protocol utilizes a hash function to convert traditional interactive zero-knowledge proofs into non-interactive ones, which not only significantly reduces communication costs but also eliminates the requirement for the verifier to be online at all times. The PSCS scheme consists of three algorithms: **PSCS-KeyGen**, **PSCS-Proof** and **PSCS-Verify**, they are constructed as follows.

- **PSCS-KeyGen** $(\lambda) \rightarrow (\mathsf{H}')$: This algorithm takes the security parameter λ as input, generates a CRS crs, and outputs a collision-resistant hash function H'.
- **PSCS-Proof** $(\mathsf{H}', w_2) \rightarrow (x_2, \pi_2)$: This algorithm takes the hash function H' as input, then generates the tuple $x_2 = (score, ct)$, $w_2 = sk_f$, where $score \in \mathbb{Z}_p$ is the credit score and ct includes the **IPFE** ciphertext components $c_0 = g^r$, $c_i = h_i^r \cdot g^{x_i}$. Then it randomly chooses $a \in \mathbb{Z}_p$ and calculates $R_i = h_i^a$, $\eta_2 = \mathsf{H}'(x_2||R_1||...||R_t||c_0||...||c_t)$ and $z_i = c_0^{s_i(\eta_2+y_i)} \cdot h_i$, $\Omega_i = c_i^{ay_i}$, $\Phi_i = h_i^{ra}$. The credit bureau finally returns the proof $x_2 = (score, ct)$ and $\pi_2 = (\eta_2, \{R_i, z_i, \Omega_i, \Phi_i\}_{i\in[1,t]})$.
- **PSCS-Verify** $(\mathsf{H}', x_2, \pi_2) \rightarrow (1/0)$: This algorithm takes the hash function H', the proof $\pi_2 = (\eta_2, R_i, z_i, \Omega_i, \Phi_i)$ and x_2 as input, then it first retrieves $\eta_2' = \mathsf{H}'(x_2||R_1||...||R_t||c_0||...||c_t)$ and checks whether $\eta_2' = \eta_2$. If not, it returns 0 to indicate the proof is invalid, otherwise it verifies the equation: $\prod_{i=1}^{t} z_i^a = \prod_{i=1}^{t}(R_i \cdot \Phi_i^{\eta_2'} \cdot \frac{\Omega_i}{g^{score}})$. If it holds, it outputs 1 to indicate the proof is valid, otherwise outputs 0.

5 Privacy-preserving credit scoring schemes

By integrating the proposed fundamental proof schemes and IPFE [11], we construct the privacy-preserving credit traditional scoring (PSCD) scheme to output the score of a credit subject. Distinguishing from traditional scoring models, we have also devised a privacy-preserving intelligent credit scoring modeling (PICS) scheme to demonstrate the implementation of scoring modeling using neural networks.

5.1 Privacy-preserving traditional credit scoring scheme

This scheme delineates the methodology employed by the credit bureau to compute the credit score of credit subject. Leveraging IPFE, without revealing any private information pertaining to the credit subject. It is noteworthy that the aforementioned constructed proof schemes have been invoked to address the trust issues between financial institutions and the credit bureau. **PPCS** scheme is consists of **PPCS-Setup**, **PPCS-ServerInitial**, **PPCS-DataEncrypt**, **PPCS-ScoreCompute** and **PPCS-ScoreVerify** algorithms and they are elaborated as follows:

- **PPCS-Setup**: KMA invokes **PSW-KeyGen**, **PSCD-KeyGen** and **PSCS-KeyGen** algorithms to obtain the public parameters $params = (p, \mathbb{G}, \mathbb{G}_T, e, g, pk, \{S_i\}_{i=0}^{l-1}, \{h_i\}_{i=1}^{l}, \mathsf{H}, \mathsf{H}')$ and private key sk. Then KMA broadcasts $params$ to the system and shares sk to the credit bureau via a secure channel.
- **PPCS-Encrypt**: The credit bureau invokes the **IPFE-KeyGen** algorithm to obtain the decryption key $sk_f = (\mathsf{y}, \mathsf{s})$. Then, it invokes the **IPFE-Encrypt** algorithm to obtain the ciphertext $c_0 = g^r$, $c_i = h_i^r \cdot g^{x_i}$ for the scoring weight k_i. It also generates the proof (x_0, π_0) for the weight k_i by running **PSW-Proof** algorithm and sends it to the financial institution, where $x_0 = (c_1, ..., c_t)$ and $\pi_0 = (\eta_0, \{V_{i,j}, a_{i,j}, \chi_i, z_{k_{i,j}}, z_{v_{i,j}}\}_{i\in[1,t], j\in[0,l-1]})$.
- **PPCS-DataEmbed**: The financial institution first verifies the validity of proof (x_0, π_0) by invoking the algorithm **PSW-Verify**. It aborts if invalid, otherwise, it retrieves $y = \sum_{i=1}^{t} g^{k_i}$, then invokes the **PSCD-Proof** algorithm to obtain the proof $x_1 = (c_1, ..., c_t)$, $\pi_1 = (h_1, \chi, \{V_{i,j}, a_{i,j}, z_{k_{i,j}}, z_{v_{i,j}}\}_{i\in[1,t], j\in[0,l-1]})$ about the credit data vector $w_1 = (m_1, ..., m_t)$, and sends them to the credit bureau.
- **PPCS-ScoreCompute**: The credit bureau invokes the **PSCD-Verify** algorithm to check the validity of proof (x_1, π_1). It aborts if invalid, otherwise, it runs the **IPFE-Decrypt** algorithm to obtain the score $score$, then invokes the **PSCS-Proof** algorithm to generate the proof (x_2, π_2) and replies it along with the score $score$ to the creditor.
- **PPCS-ScoreVerify**: The creditor verifies the validity of proof (x_2, π_2) by invoking **PSCS-Verify** algorithm, if it is valid, the creditor accepts the credit score $score$, otherwise rejects it.

5.2 Privacy-preserving intelligent credit scoring modeling scheme

Apart from traditional scoring models, some proposed studies use machine learning for scoring modeling, achieving more precise scoring results. Here, we primarily focus on constructing a privacy-preserving intelligent scoring modeling system based on BP neural networks. The computation within a neural network are predominantly vector operations of activation functions within each neuron, and it can be realized in a ciphertext state based on the IPFE protocol. Compared to traditional scoring models, the credit scoring model based on BP neural networks no longer rely on fixed weights. Instead, the weights adaptively adjust according to the loss value to achieve the best fit. Therefore, we no longer require a credit bureau to certify the range of weights to financial institutions. Since there is no linear relationship between the ciphertext of a credit subject's data and the credit score in a BP neural network, the proof between ciphertext and score is also meaningless. Here, we describes how to use functional encryption for privacy-preserving scoring modeling based on neural networks by the following privacy-preserving intelligent credit scoring (PICS) scheme.

• **PICS-Setup**: KMA invokes **PSW-KeyGen** and **PSCD-KeyGen** algorithms to obtain the public parameters $params$ and private key sk. Then KMA broadcasts $params$ to the system and shares sk to the credit bureau via a secure channel.

• **PICS-KeyGen**: The credit bureau invokes **IPFE-KeyGen** algorithms to obtain the decryption key sk_f about the initial weight $w = (k_1, ..., k_t)$. The credit bureau also runs **PSW-Proof** to generate a proof (x_0, π_0) for the initial weight w, which is blind to the financial institution.

• **PICS-Encrypt**: The financial institution first executes **PSW-Verify** algorithm to check the validity of proof (x_0, π_0), if it is valid, then invokes the **IPFE-Encrypt** algorithm to encrypt its input vector $\mathsf{x} = (x_1, ..., x_t)$, then outputs and returns $c_0, \{c_i\}$ to the credit bureau for each x_i, where $i \in [1, t]$. The financial institution also generates the proof (x_1, π_1) of the ciphertext $c_0, \{c_i\}$ by invoking **PSCD-Proof** algorithm.

• **PICS-Train**: The credit bureau first verifies the validity of proof (x_1, π_1) by running **PSCD-Verify** algorithm, if it is valid, then the credit bureau invokes the **IPFE-Decrypt** algorithm to decrypt the ciphertext c_i and obtain the inner product $prod_i$. In other words, the activation function is calculated in the ciphertext-form $sk_f(w^T) \cdot$ **IPFE-Encrypt**(x). Then, it returns the value of activation function $g(prod_i + b_i)$, and it will serve as input to the next layer of the neural network. Finally, the credit bureau returns the score to the creditor.

6 Security analysis

We rigorously demonstrate the IND-CPA security of the IPFE scheme employed in our protocol, as well as the indistinguishability of the PPCS and PICS schemes with respect to weights and data. Specifically, we reduce the IND-CPA security of IPFE to the Diffie-Hellman Decisional (DDH) problem by constructing a polynomial-time simulator $\mathcal{C}$, which interacts with an adversary $\mathcal{A}$. If $\mathcal{A}$, with the assistance of $\mathcal{C}$, can solve the DDH problem, $\mathcal{A}$ wins, and hence the IND-CPA security of IPFE is compromised. However, given that the DDH problem is computationally hard, the probability of it being solved is negligible. For PPCS and PICS, we similarly construct two simulators, $S0$ and $S1$, which simulate proofs regarding data and weights using randomly selected resources (rather than valid ones). If these proofs can pass the verification algorithm, the confidentiality of the data and weights is breached. The detailed proofs are presented as follows:

Theorem 1 *IPFE scheme used in the proposed schemes is indistinguishability under the chosen plaintext attack (IND-CPA) if the DDH assumption holds.*

Proof Assume there is an adversary $\mathcal{A}$ capable of compromising the IND-FE-CPA security of the IPFE scheme with a non-negligible advantage. Subsequently, we devise a challenge simulator $\mathcal{C}$ that, given an instance of the DDH assumption (g, g^a, g^b, g^c), where c could be either ab or a random value, leverages $\mathcal{C}$ to break this assumption.

In cases where the challenge messages x_0 and x_1 differ, there exists a message space vector for which the adversary should remain unaware of the key $(\mathsf{x}_1 - \mathsf{x}_0$ represents one such vector).

To create the master public key, $\mathcal{C}$ first generates secret keys for a basis z_i derived from $(\mathsf{x}_1 - \mathsf{x}_0)$. By implicitly setting a as the secret key for $\mathsf{x}_1 - \mathsf{x}_0$, $\mathcal{C}$ utilizes g^a to produce the master public key. Once group elements for the basis y_i, augmented with $(\mathsf{x}_1 - \mathsf{x}_0)$, are generated, the public key for the canonical basis can be determined. For generating the challenge ciphertext, $\mathcal{C}$ selects a random bit and employs g^b and g^c to create a ciphertext corresponding to message x. It is important to note that, per the rules of the IND-FE-CPA security game $\mathsf{Game}_{IND-FE-CPA}$, $\mathcal{A}$ is permitted to inquire only about secret keys for vectors within the subspace generated by the z_i's, which are orthogonal to $\mathsf{x}_1 - \mathsf{x}_0$. For these vectors, $\mathcal{C}$ can produce the corresponding secret keys.

If $c = ab$, the resulting challenge ciphertext corresponds to a well-distributed encryption of the message x_μ. Conversely, when c is random, the ciphertext instead represents an encryption of the message $u\mathsf{x}_0 + (1 - u)\mathsf{x}_1$, where u is uniformly distributed. In this scenario, the message remains information-theoretically concealed. This is because, for any decryption key sk_y query, y being orthogonal to $\mathsf{x}_1 - \mathsf{x}_0$ ensures $\langle \mathsf{x}_0, \mathsf{y} \rangle = \langle \mathsf{x}_1, \mathsf{y} \rangle$, while the decrypted ciphertexts match their respective inner products $\langle u\mathsf{x}_0 + (1 - u)\mathsf{x}_1 \rangle = u\langle \mathsf{x}_0, \mathsf{y} \rangle + (1 - u)\langle \mathsf{x}_1, \mathsf{y} \rangle = u\langle \mathsf{x}_\mu, \mathsf{y} \rangle + (1 - u)\langle \mathsf{x}_\mu, \mathsf{y} \rangle = \langle \mathsf{x}_\mu, \mathsf{y} \rangle$.(Please see [11] for details of the proof.) □

Theorem 2 *The proposed schemes satisfy weight and credit data confidentiality.*

Proof Since both the PPCS and PICS schemes invoke the PSW and PSCD schemes, we focus the proof of confidentiality on the weight and credit data. Following the proof methodology outlined in [26], we need to construct two simulators, $S0$ and $S1$, capable of simulating the proofs π_0 and π_1 respectively, under conditions where the necessary resources are absent.

Simulator $S0$: It first invokes **PSW-KeyGen** algorithm to obtain the public key pk, private key sk and public parameters $pp = (p, \mathbb{G}, \mathbb{G}_T, g, e, pk, \{S_i\}_{i=1}^{t}, \mathsf{H})$, then given the random tuple $x_0 = (c_1, ..., c_t)$. Instead of invoking the **PSW-Proof** algorithm, it randomly chooses $\eta_0^* \in \{0, 1\}^l$, and also selects $v_{i,j}^*, z_{k_{i,j}}^*, z_{v_{i,j}}^* \in \mathbb{Z}_p$, then calculates $V_{i,j}^* = S_{k_{i,j}}^{v_{i,j}^*}$, $\chi_i^* = \prod_{j=0}^{l-1}(g^{(z_{k_{i,j}}^* + k_{i,j}\eta_0)u_j}) = c_i^{\eta_0^*}\prod_{j=0}^{l-1}(g^{z_{k_{i,j}}^* u_j})$ and $a_{i,j}^* = e(V_{i,j}, y)^{\eta_0^*} \cdot e(V_{i,j}, g)^{-z_{k_{i,j}}^*} \cdot e(g, g)^{z_{v_{i,j}}^*}$. Define a random oracle such that $\mathcal{O}(V_{i,j}^* || a_{i,j}^* || \chi_i^*) = \eta_0^*$, where $i \in [1, t]$, $j \in [0, l-1]$. S_0 finally returns the simulated proof $\pi_0^* = (\eta_0^*, \{V_{i,j}^*, a_{i,j}^*, \chi_i^*, z_{k_{i,j}}^*, z_{v_{i,j}}^*\}_{i\in[1,t], j\in[0,l-1]})$.

Simulator $S1$: It first invokes **PSCD-KeyGen** algorithm to obtain the private key (sk, s) and public parameters $pp = (p, \mathbb{G}, \mathbb{G}_T, g, e, pk, \{S_i\}_{i=1}^{t}, \{h_i\}_{i=1}^{t}, \mathsf{H})$, then given the random tuple $x_1 = (c_1, ..., c_t)$. Instead of invoking the **PSCD-Proof** algorithm, it randomly chooses $\eta_1^* \in \{0, 1\}^l$, and also selects $v_{i,j}^*, z_{m_{i,j}}^*, z_{v_{i,j}}^*, z_{h_i}^* \in \mathbb{Z}_p$, then calculates $V_{i,j}^* = S_{m_{i,j}}^{v_{i,j}^*}$, $\chi_i^* = \prod_{j=1}^{l-1}(g^{u^j(z_{m_{i,j}}^* + m_{i,j}\eta_1^*)}) = c_i^{-\eta_1^*} \cdot \prod_{j=0}^{l-1} g^{u^j z_{m_{i,j}^*}} \cdot z_{h_i}^*$ and $a_{i,j}^* = e(V_{i,j}, y)^{\eta_1^*} \cdot e(V_{i,j}, g)^{-z_{k_{i,j}}^*} \cdot e(g, g)^{z_{v_{i,j}}^*}$. Define a random oracle such that $\mathcal{O}(V_{i,j}^* || a_{i,j}^* || \chi_i^*) = \eta_1^*$, where $i \in [1, t]$, $j \in [0, l-1]$. S_1 finally returns the simulated proof $\pi_1^* = (\eta_1^*, \{V_{i,j}^*, a_{i,j}^*, \chi_i^*, z_{k_{i,j}}^*, z_{v_{i,j}}^*, z_{h_i}^*\}_{i\in[1,t], j\in[0,l-1]})$. □

7 Performance evaluations

This section evaluates the performance of the proposed PPCS and PICS security scoring schemes, encompassing time overhead, prediction accuracy of risk scores, and scoring effectiveness. Additionally, a comparison of the proposed schemes with relevant representative works is conducted.

The testbench is a PC configured with an AMD Ryzen 5 4600H CPU, 16 GB of RAM, and 64-bit Windows 10 OS. The experiment was conducted to construct neural networks using Python-3.9.7 and the CPU version of PyTorch-1.11.0, and the cryptographic operations were implemented based on the PyCryptodome, Pypbc and Gmpy2 libraries. Based on the aforementioned configuration, the evaluations of computational and storage performance, as well as scoring performance, are presented below.

7.1 Evaluation of computational and storage performance

The storage and computational complexity statistics of the three fundamental schemes (PSW,PSCD and PSCS) proposed in this paper are shown in Table 1. For the storage complexities, we evaluate the public parameters and the proof, while for the computational complexities, we evaluate the public parameters generation, proof generation and proof verification.

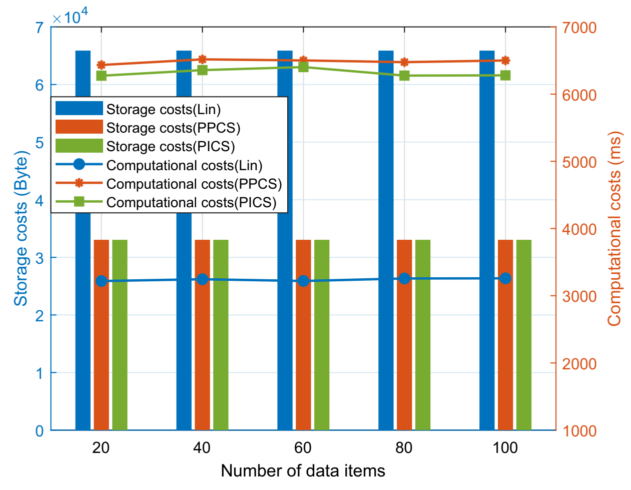
It is evident that the storage and computational complexity of public parameters and proofs in the aforementioned schemes are mostly related to the length of vectors. Since both PSW and PSCD are constructed based on the Camenisch et al.'s Range Proof protocol [21], their computational and storage costs are generally associated with the exponent of the upper bound of the range. In contrast, the storage and computational complexity of PSCS are significantly lower than those of PSW and PSCD.

Subsequently, we implemented both the PPCS scheme and Lin et al.'s [26] scheme, and conducted a comparison of their communication and computational costs at various phases. To ensure consistency, we set the maximum length of weights to 30 bits, thereby limiting the range of weight values to $[0, 2^{10})$ and $l = 3$. To attain 80-bit security, we choose to implement the scheme on a supersingular elliptic curve of the form $y^2 = x^3 + x$ with a degree of 2^{160}. Then we obtain $|\mathbb{G}| = |\mathbb{G}_T| = 1024\text{bit} = 128\text{byte}$ and $|\mathbb{Z}_p| = 160\text{bit} = 20\text{byte}$, while the parameter setting of Lin et al.'s scheme is consistent with the original paper [26]. Under the aforementioned settings, the comparison of PPCS, PICS, and Lin et al.'s schemes in terms of storage and computational costs is illustrated in Fig. 3. Figure 3-(a)

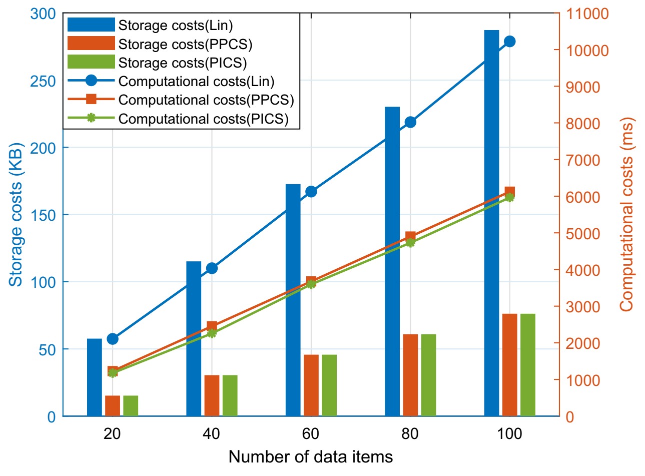
Table 1 Storage and computational complexities of fundamental schemes

Schemes		PSW	PSCD	PSCS
Storage complexities	public parameters	$(u+2)\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_p\rvert$	$(2u+2)\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_p\rvert$	$\lvert\mathbb{Z}_p\rvert$
	proof	$2t\lvert\mathbb{G}\rvert + tl\lvert\mathbb{G}_T\rvert + (2tl+t+1)\lvert\mathbb{Z}_p\rvert$	$(tl+3t+1)\lvert\mathbb{G}\rvert + tl\lvert\mathbb{G}_T\rvert + tl\lvert\mathbb{Z}_p\rvert$	$(5t+2)\lvert\mathbb{G}\rvert$
Computational complexities	public parameters	$(u+1)e_G$	$(2u+1)e_G$	N/A
	proof generation	$2tl\mathbb{P} + 3tle_{G_T} + (tl+2t)e_G + H$	$tl\mathbb{P} + 2tle_{G_T} + (3tl+2t)e_G + H$	$(6t+1)e_G + H$
	proof varification	$2tl\mathbb{P} + 3tle_{G_T} + (tl+2t)e_G + H$	$2tl\mathbb{P} + 3tle_{G_T} + (tl+t+1)e_G + H$	$2te_G + H$

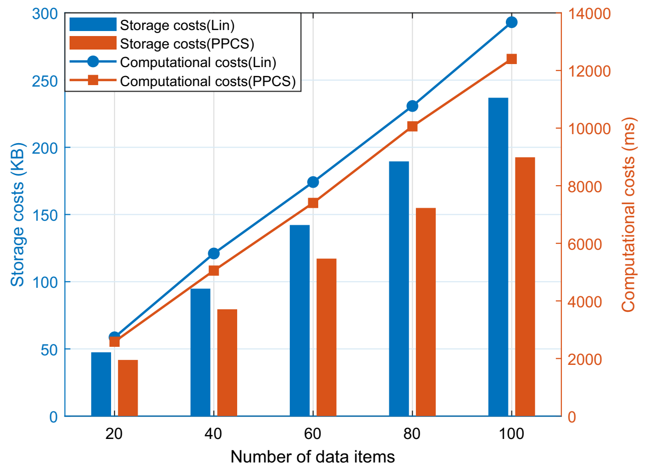
Notations: $|\mathbb{G}|$: size of an element in group $\mathbb{G}$; $|\mathbb{G}_T|$: size of an element in group $\mathbb{G}_T$; $|\mathbb{Z}_p|$: size of an element in group $\mathbb{Z}_p$; t: length of the vectors; l: exponent of the upper range bound; u: base number of the upper range bound; $\mathbb{P}$: a pairing operation; e_G: a modular exponentiation over group $\mathbb{G}$; e_{G_T}: a modular exponentiation over group $\mathbb{G}_T$; H: a hash operation



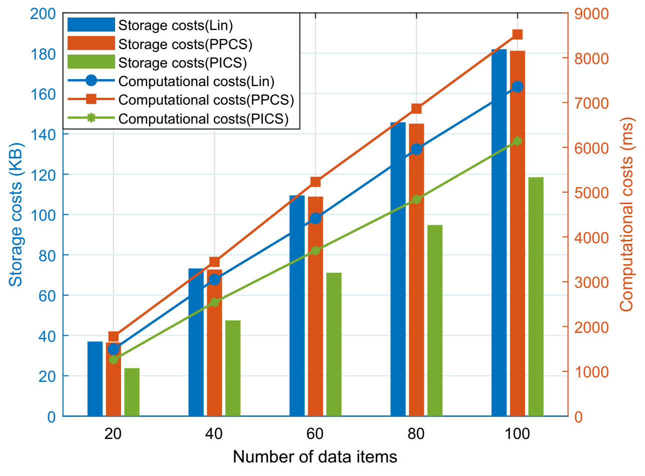
(a) Setup&Keygen



(b) Encrypt



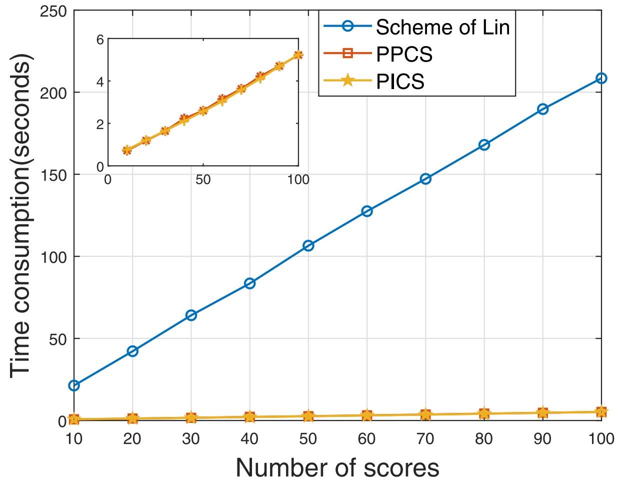
(c) DataEmbed



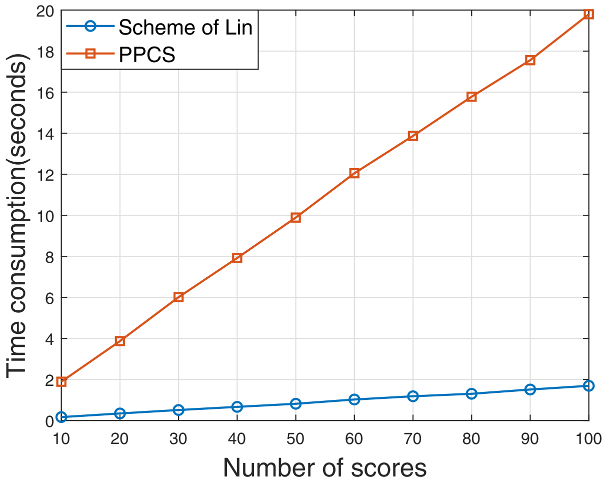
(d) ScoreExtract&Verify&Train

Fig. 3 Comparisons of computational costs. *Please note that, due to the significant algorithmic structural differences between PICS and PPCS, Lin's scheme, for the sake of fairness, this figure solely compares the storage and computational costs of similar logical operations. The model training time cost of PICS has not been counted in Fig. 3-(d)

demonstrates that during the Setup&Keygen phase, as the number of data items (length of the weight vectors and data vectors) increases, the computational and storage costs of all schemes remain approximately constant. Among them, PPCS and PICS exhibit lower storage costs compared to Lin et al.'s scheme, whereas Lin et al.' scheme outperform PPCS and PICS in terms of computational cost. Figure 3-(b) shows that during the Encrypt phase, the computational and storage costs of all schemes increase approximately linearly with the number of data items. Notably, PPCS and PICS outperform Lin et al.'s scheme in both storage and computational costs. A significant reason for this is that PPCS and PICS utilize inner product function encryption instead of Paillier homomorphic encryption, resulting in a significant improvement in computational efficiency. When the number of data items reaches 100, PPCS and PICS only take 6123.14ms and 5963.57 ms, respectively, representing performance improvements of 40.1% and 41.7% compared to Lin et al.'s scheme. During the DataEmbed&Encrypt phase, as illustrated in Fig. 3-(c), the storage and computational costs of all schemes exhibit an approximately linear growth trend. PPCS outperforms Lin et al.'s scheme, saving 9.3% of the time when the number of data items reaches 100. In the ScoreExtract&Verify&Train phase, all schemes continue to increase with the number of data items. Due to the involvement of neural network model training in PICS at this phase,



(a) Data encryption



(b) Score extraction

Fig. 4 Computational time cost versus number of tasks

for fairness, the time for model training is not included in the computational cost. Given that PICS incorporates neural network model training at this phase, for the sake of fairness, the duration of model training has been excluded from the computational cost analysis. In the case of PPCS, while its storage cost is marginally lower compared to Lin et al.'s scheme, its computational cost is 15.9% higher. This disparity arises because PPCS conducts the final score calculation within this phase.

the aforementioned schemes after verification through proof. The evaluation results are presented in Fig. 4. The range of scoring tasks was set from 10 to 100. From Fig. 4-(a), it is evident that PPCS and PICS outperform Lin et al.'s scheme significantly in terms of data encryption performance. Furthermore, as the task scale increases, the performance advantages of functional encryption become more pronounced. When processing 100 scoring tasks, PPCS, PICS, and Lin et al.'s scheme consumed 208.57s, 5.22s, and 5.29s, respectively. Figure 4-(b) indicates that Lin et al.'s scheme exhibits superior performance in score extraction compared to PPCS, consuming 1.691s and 19.810s, respectively, for extracting 100 scores. Please note that it

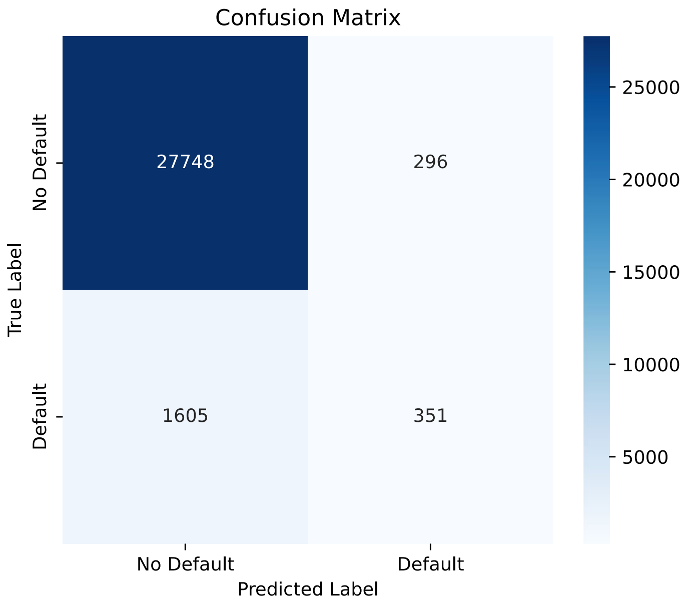


Fig. 5 Confusion matrix of the binary predication model

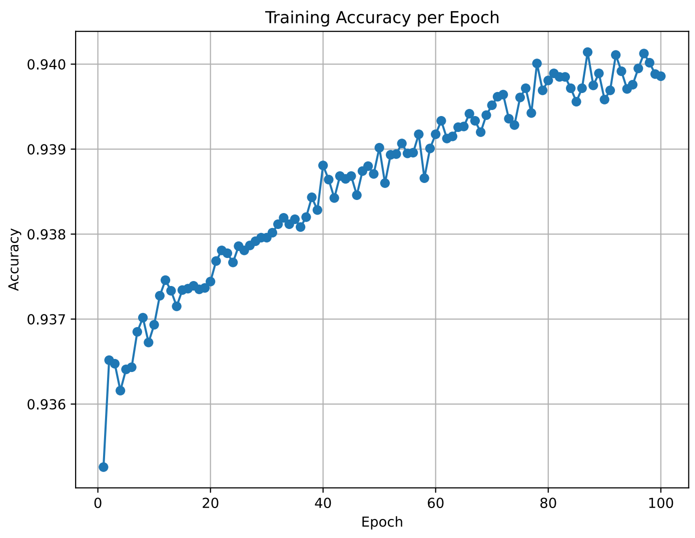


Fig. 6 Accuracy of the binary predication model

is unreasonable to directly include PICS in the comparison of time costs for score extraction, as PICS's score extraction relies on the training of neural network models.

7.2 Evaluation of scoring performance

In addition to evaluating computational performance, this experiment also assessed the effectiveness of the PICS scheme in terms of score prediction. Due to its exceptional self-learning and optimization capabilities, we have utilized the BP neural network as the fundamental model. The experiments are based on the UCI_Credit_Card dataset [28], which is publicly available and frequently used in credit assessment research. The UCI_Credit_Card dataset collects credit card lending data in 2005, covering items such as credit limit, marital status, age, education level, and monthly repayment status from April to September, 2005. The training set contains 150,000 records, while the test set exceeds 100,000 records, this experiment does not use the test set because it does not provide labels for default behavior within the next two years. The proposed BP neural network consists of an input layer, a hidden layer (which comprises two sub-layers), and an output layer. Specifically, the first sub-layer of the hidden layer maps the 64 features from the input layer to 32 features, while the second sub-layer maps these 32 features to a single output in the output layer.

Initially, we employed a BP neural network for binary risk prediction, specifically to forecast whether a credit entity would experience a severe default within the following two years. Given that the test set of UCI_Credit_Card does not

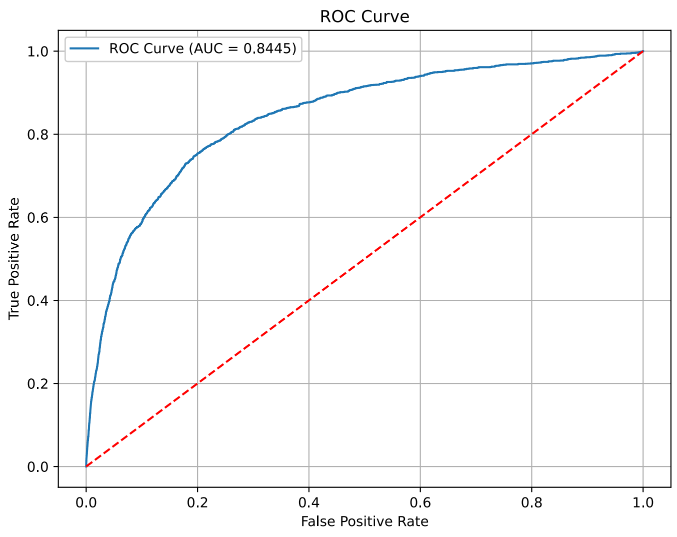


Fig. 7 ROC of the binary predication model

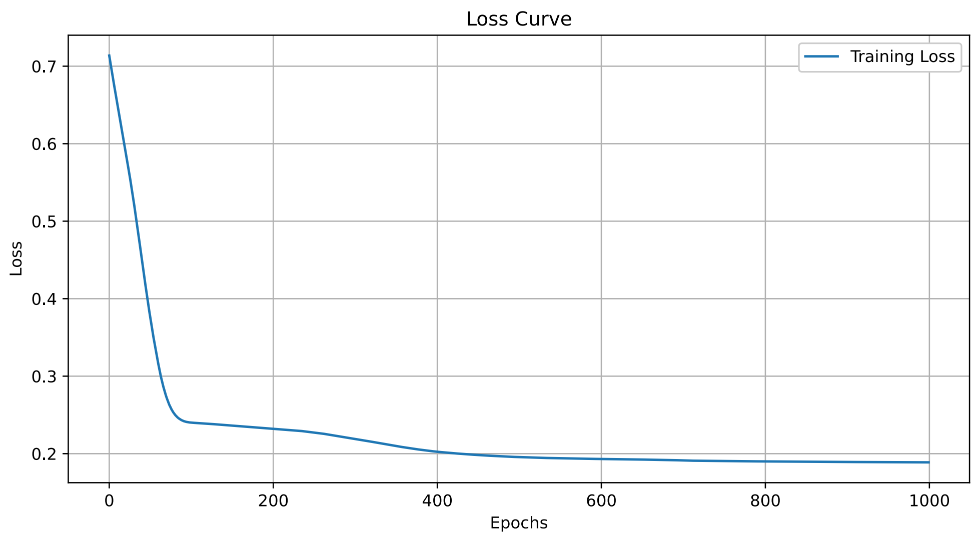


Fig. 8 Loss of each epoch for the scoring predication model

contain default information, we designated the first 120,000 rows of the training set as the training dataset and the subsequent 30,000 rows as the test dataset. With 100 epochs of training, the confusion matrix for the prediction results of the binary classification model is depicted in Fig. 5. Based on this matrix, we obtained an accuracy of 94.0% (as illustrated in Fig. 6), a precision of 54.3%, a recall rate of 17.9%, and an F1 score of 19.0% [33].

Figure 7 displays the ROC curve for the binary classification model, with an AUC value approaching 0.85, demonstrating the excellent discriminatory capability of the proposed binary classification prediction model based on the BP neural network.

However, the UCI_Credit_Card training dataset only provides binary classification data indicating whether a credit entity will experience a serious default within the next two years, posing challenges for credit score prediction. For instance, Li et al. [33] proposed a federated learning credit evaluation scheme based on functional encryption, but due to limitations in publicly available datasets, it was also limited to binary classification predictions. In response to this, initially we constructed a model utilizing a Backpropagation (BP) neural network to forecast the default probability of credit entities and mapped these probabilities onto the FICO score range of [300,850] [36] in descending order of default probability. The model underwent training for 1000 epochs, with the decreasing trend of the loss function shown in Fig. 8. Following this, the statistical distribution of scores for both the training and testing datasets is presented in Fig. 9. The results indicate a strong alignment between the predicted score distribution of the testing dataset and that of the training dataset, with the majority of predicted scores falling within the range

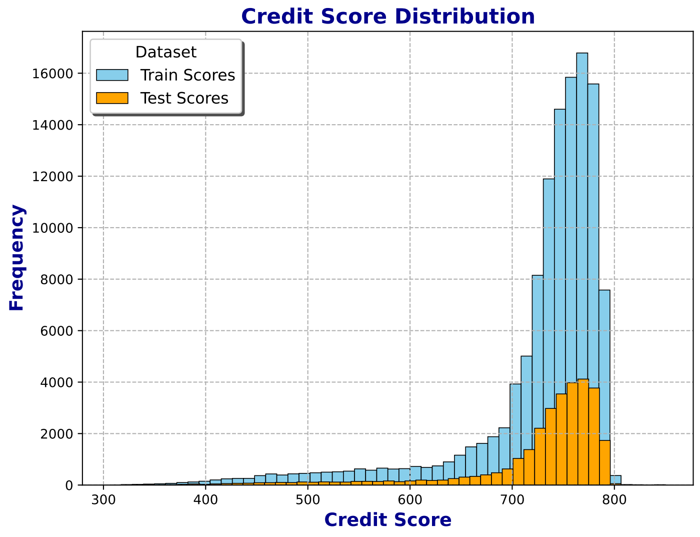


Fig. 9 Distribution of the credit score

of 600-800. This demonstrates the promising predictive performance of the scoring prediction model constructed in this study.

8 Conclusion

The credit reporting system generates credit scores through the processing of credit data, serving as a “permit” for the financial activities of enterprises or individuals, and playing an increasingly crucial role in people’s financial endeavors. To address the security, privacy, and multi-party trust issues in the optimization process of credit scoring, this paper first constructs three proof schemes. Based on these, a privacy-preserving credit scoring scheme (PPCS) and a privacy-preserving intelligent credit scoring scheme (PICS) are designed. PPCS employs inner-product functional encryption (IPFE) instead of homomorphic encryption to achieve efficient encrypted scoring calculations, and utilizes the proposed proof schemes to verify the ranges of weights and credit data, as well as the association between ciphertexts and scores. PICS, on the other hand, only provides proof of the range for the initial weights in deep learning and uses IPFE to implement encrypted vector operations during neural network training. Subsequently, we demonstrate the security of the schemes and construct experiments to prove the superiority of PPCS and PICS in terms of computational and storage efficiency, as well as the excellent performance of PICS in binary classification prediction and credit score prediction.

Author Contributions Yangyang Bao: Modelling, Designing, and Writing; Lingrui Pan: Implementing; Xiaochun Cheng: Supervision and proofreading; Liming Nie: Methodology and supervision.

Funding This paper is sponsored by China Postdoctoral Science Foundation (Grant Number: 2024M753597); National Natural Science Foundation of China (NSFC) (Grant Number: W2412110); UKRI EPSRC Grant funded Doctoral Training Centre at Swansea University, through PhD project RS718 on Explainable AI; UKRI EPSRC Grant EP/W020408/1 Project SPRITE+ 2: The Security, Privacy, Identity and Trust Engagement Network plus (phase 2); PhD project RS718 on Explainable AI through UKRI EPSRC Grant funded Doctoral Training Centre at Swansea University.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

Ethics approval and consent to participate The work is original and unpublished and has not been submitted for publication elsewhere.

Consent for publication All authors agree to the publication.

References

1. Bradford T (2023) Give me some credit!: using alternative data to expand credit access. Payments system research briefing, pp 1–6
2. Hassija V, Bansal G, Chamola V et al (2020) Secure lending: blockchain and prospect theory-based decentralized credit scoring model. IEEE Trans Netw Sci Eng 7(4):2566–2575
3. Dowd AC, Rosinger KO, Fernandez Castro M (2020) Trends and perspectives on finance equity and the promise of community colleges. Higher education: handbook of theory and research, vol 35, pp 517–588
4. Moscato V, Picariello A, Sperlì G (2021) A benchmark of machine learning approaches for credit score prediction. Expert Syst Appl 165:113986
5. Yang F, Abedin MZ, Hajek P (2024) An explainable federated learning and blockchain-based secure credit modeling method. Eur J Oper Res 317(2):449–467
6. Ferreira M, Brito T, Santos JF et al (2023) RuleKeeper: GDPR-aware personal data compliance for web frameworks. 2023 IEEE Symposium on Security and Privacy (SP). IEEE, pp 2817–2834
7. Hijazi NM, Aloqaily M, Guizani M et al (2023) Secure federated learning with fully homomorphic encryption for iot communications. IEEE Internet Things J 11(3):4289–4300
8. Al-Turjman F, Baali I (2022) Machine learning for wearable IoT-based applications: a survey. Trans Emerg Telecommun Technol 33(8):e3635
9. Brevoort KP, Grimm P, Kambara M (2016) Credit invisibles and the unscored. Cityscape 18(2):9–34
10. Andolfo L, Coppolino L, D Antonio S et al (2021) Privacy-preserving credit scoring via functional encryption, Computational science and its applications–CICCSA 2021: 21st International Conference, Cagliari, Italy, September 13–C16, 2021, Proceedings, Part VIII 21. Springer International Publishing, LNTCS, vol 12956, pp 31–43
11. Abdalla M, Bourse F, De Caro A et al (2015) Simple functional encryption schemes for inner products. IACR international workshop on public key cryptography. Berlin, Heidelberg: Springer, Berlin Heidelberg, LNSC, vol 9020, pp 733–751
12. Lee JW, Kang HC, Lee Y et al (2022) Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. IEEE Access 10:30039–30054
13. Zapechnikov S (2022) Secure multi-party computations for privacy-preserving machine learning. Procedia Comput Sci 213:523–527
14. Wang H, Wang Q, Ding Y et al (2024) Privacy-preserving federated learning based on partial low-quality data. J Cloud Comput 13(1):62
15. Li X, He J, Vijayakumar P et al (2021) A verifiable privacy-preserving machine learning prediction scheme for edge-enhanced HCPSs. IEEE Trans Ind Inform 18(8):5494–5503
16. Xu G, Li H, Liu S et al (2019) VerifyNet: secure and verifiable federated learning. IEEE Trans Inf Forensic Secur 15:911–926
17. Pang LJ, Wang YM (2005) A new (t, n)-multi-secret sharing scheme based on Shamir’s secret sharing. Appl Math Comput 167(2):840–848
18. Xu R, Joshi JB, Li C (2019) Cryptonn: training neural networks over encrypted data. IEEE 39th International Conference on Distributed Computing Systems (ICDCS) 2019, pp 1199–1209
19. Panzade P, Takabi D (2023) FENet: privacy-preserving neural network training with functional encryption. Proceedings of the 9th ACM international workshop on security and privacy analytics, pp 33–43
20. Ryffel T, Dufour-Sans E, Gay R et al (2019) Partially encrypted machine learning using functional encryption. arXiv:1905.10214

21. Camenisch J, Chaabouni R, Shelat A (2008) Efficient protocols for set membership and range proofs. International conference on the theory and application of cryptology and information security. Berlin, Heidelberg: Springer Berlin Heidelberg, LNSC, vol 5350, pp 234–252
22. Hamila F, Hamad M, Salgado DC et al (2024) Enhancing security in Fiat-CShamir transformation-based non-interactive zero-knowledge protocols for IoT authentication. Int J Inf Secur 23(2):1131–1148
23. Deng Y, Ma S, Zhang X et al (2021) Promise Σ-protocol: how to construct efficient threshold ECDSA from encryptions based on class groups. International conference on the theory and application of cryptology and information security. Cham: Springer International Publishing, LNSC, vol 13093, pp 557–586
24. Boneh D, Boyen X (2004) Efficient selective-ID secure identity-based encryption without random oracles. Advances in cryptology-EUROCRYPT 2004: international conference on the theory and applications of cryptographic techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23. Springer Berlin Heidelberg, LNSC, vol 3027, pp 223–238
25. Canetti R, Chen Y, Holmgren J et al (2019) Fiat-Shamir: from practice to theory. Proceedings of the 51st annual ACM SIGACT symposium on theory of computing, pp 1082–1090
26. Lin C, Luo M, Huang X et al (2021) An efficient privacy-preserving credit score system based on noninteractive zero-knowledge proof. IEEE Syst J 16(1):1592–1601
27. Naresh VS (2024) PPDNN-CRP: privacy-preserving deep neural network processing for credit risk prediction in cloud: a homomorphic encryption-based approach. J Cloud Comput 13(1):149
28. UCI_credit_card. https://www.kaggle.com/datasets/xuandiluo/uci-credit-cardcsv
29. Qiao Y, Lan Q, Zhou Z et al (2022) Privacy-preserving credit evaluation system based on blockchain. Expert Syst Appl 188:115989
30. Qiao Y, Lan Q, Wang Y et al (2023) PEvaChain: privacy-preserving ridge regression-based credit evaluation system using hyperledger fabric blockchain. Expert Syst Appl 223:119844
31. Noh H, Ji S, Go Y et al (2024) Resilient and Fast Block Transmission System for Scalable Hyperledger Fabric Blockchain in Multi-Cloud Environments. IEEE Trans Netw Serv Manag 21(5):5118–5134
32. Sun J, Bao Y, Qiu W et al (2024) Privacy-preserving fine-grained data sharing with dynamic service for the cloud-edge IoT. IEEE Trans Dependable Secure Comput 1–18. https://doi.org/10.1109/TDSC.2024.3432650
33. Li W, Wang S, Song Y (2024) Personal credit evaluation model based on federated learning. 2024 IEEE 14th International Conference on Electronics Information and Emergency Communication (ICEIEC). IEEE, pp 1–7
34. He H, Wang Z, Jain H et al (2023) A privacy-preserving decentralized credit scoring method based on multi-party information. Decis Support Syst 166:113910
35. Jovanovic Z, Hou Z, Biswas K et al (2024) Robust integration of blockchain and explainable federated learning for automated credit scoring. Comput Netw 243:110303
36. Zakowska A (2023) A new credit scoring model to reduce potential predatory lending: a design science approach. International conference on systems engineering. Cham: Springer Nature Switzerland, LNNS, vol 761, pp 33–47
37. Panzade P, Takabi D, Cai Z (2024) Privacy-preserving machine learning using functional encryption: opportunities and challenges. IEEE Internet Things J 11(5):7436–7446

Yangyang Bao received the B.S. and M.S. degree in software engineering from the School of Information and Software Engineering, University of Electronic Science and Technology of China, China, in 2016 and 2019, and also received the Ph.D. degree in information security from the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China, in 2023. He is currently a postdoctoral researcher with the Shanghai Jiao Tong University, China, and the Credit Reference Center of People's Bank of China, China. His research interests include cryptography, privacy computing, network security, machine learning and financial risk management.

Lingrui Pan received the B.S. and M.S. degree in software engineering from Xi'an Jiao Tong Univeristy, China, in 2014 and 2017. He is currently a researcher with the Credit Reference Center of People's Bank of China, China. His research interests include financial risk management.

Xiaochun Cheng received the B.Eng. degree in computer software engineering and the Ph.D. degree in computer science from Jilin University, Changchun, China, in 1992 and 1996, respectively. He has been the professor in Swansea University, Wales, U.K., since 2022. Dr. Cheng is a member of IEEE SMC Technical Committee on Enterprise Information Systems, IEEE SMC Technical Committee on Computational Intelligence, IEEE SMC Technical Committee on Cognitive Computing, IEEE SMC Technical Committee on Intelligent Internet Systems, IEEE Communications Society Communications, and Information Security Technical Committee, BCS Information Security Specialist Group, BCS Cybercrime Forensics Specialist Group, and BCS Artificial Intelligence Specialist Group.

Liming Nie received the Ph.D. degree from the Dalian University of Technology in 2017. He subsequently joined Zhejiang Sci-Tech University as a Faculty Member. He was a Senior Research Fellow with Nanyang Technological University since 2021. And he is current the associate professor with Shenzhen Technology University. His current research interests include intelligent software development, big code data analysis, and profiling open-source software.

Authors and Affiliations

Yangyang Bao[1,5] · Lingrui Pan[2] · Xiaochun Cheng[3] · Liming Nie[4]

✉ Xiaochun Cheng
xiaochun.cheng@swansea.ac.uk

✉ Liming Nie
nieliming@sztu.edu.cn

Yangyang Bao
byy_sjtu@sjtu.edu.cn

Lingrui Pan
plingrui@crcmir.org.cn

[1] Antai college of economic and management, Shanghai Jiao Tong University, 1954 Huashan Road, Xuhui District, Shanghai Municipality 200230, China

[2] Credit reference center, the People's Bank of China, 298 Fanchang Road, Pudong New Area, Shanghai Municipality 201201, China

[3] Department of Computer Science, Swansea University, Bay Campus, Fabian Way, Swansea SA1 8EN, Wales, U.K.

[4] College of Big Data and Internet, Shenzhen Technology University, Shenzhen 518118, Guangdong Province, China

[5] Postdoctoral workstation of Credit reference center, the People's Bank of China, 298 Fanchang Road, Pudong New Area, Shanghai Municipality 201201, China