Hindawi
Security and Communication Networks
Volume 2020, Article ID 8872853, 10 pages
https://doi.org/10.1155/2020/8872853

Research Article

Privacy-Preserving Blockchain-Based Nonlinear SVM Classifier Training for Social Networks

Nan Jia,[1] **Shaojing Fu**,[1,2] **and Ming Xu**[1]

[1]*College of Computer, National University of Defense Technology, Changsha, China*
[2]*State Key Laboratory of Cryptology, Beijing, China*

Correspondence should be addressed to Shaojing Fu; shaojing1984@163.com

Received 10 September 2020; Revised 3 November 2020; Accepted 25 November 2020; Published 14 December 2020

Academic Editor: Ilsun You

With the development of social networks, there are more and more social data produced, which usually contain valuable knowledge that can be utilized in many fields, such as commodity recommendation and sentimental analysis. The SVM classifier, as one of the most prevailing machine learning techniques for classification, is a crucial tool for social data analysis. Since training a high-quality SVM classifier usually requires a huge amount of data, it is a better choice for individuals and small enterprises to conduct collaborative training with multiple parties. Nevertheless, it causes privacy risks when sharing sensitive data with untrusted people and enterprises. Existing solutions mainly adopt the computation-intensive cryptographic methods which are not efficient for practical applications. Therefore, it is an urgent and challenging task to realize efficient SVM classifier training while protecting privacy. In this paper, we propose a novel privacy-preserving nonlinear SVM classifier training scheme based on blockchain. We first design a series of secure computation protocols which can achieve secure nonlinear SVM classifier training with minimal computation overheads. Then, leveraging these building blocks, we propose a blockchain-based secure nonlinear SVM classifier training scheme that realizes collaborative training while protecting privacy. We conduct a thorough analysis of the security properties of our scheme. Experiments over a real dataset show that our scheme achieves high accuracy and practical efficiency.

1. Introduction

Nowadays, social networks have been playing a significant role in reflecting and influencing human living styles. It makes people be able to keep in touch with each other and share information anytime and anywhere. The development of social networks has resulted in more and more social-related data being produced, which consist of various raw insights and information. With the evolution of artificial intelligence and machine learning, many individuals and companies try to train learning models by utilizing social data to learn valuable information for personal or commercial purposes. Support vector machine (SVM), as one of the most essential and important machine learning techniques for classification, can be applied to many fields, such as medical diagnosis [1], image recognition [2], and recommendation system [3]. It is also crucial for social data analysis and processing. For instance, many e-commerce companies may collect social information about potential customers and train effective SVM classifiers to conduct commodity recommendation. In addition, individuals may want to train SVM models by utilizing photos from social media to support automated image annotation, which can make it more convenient to find photos from electronic albums. To obtain an SVM classifier with high accuracy, the training process usually requires a huge amount of social data. However, it is difficult for a single party (e.g., an individual or a company) to collect plentiful and diversified data. Therefore, there has been a surge in demand for collaborative training by a group of parties. The merged dataset from multiple sources on social networks has obvious advantages on data volume and variety. Therefore, it is a growing trend for companies and individuals to share information and collaboratively train a high-quality

classifier. However, it causes nonnegligible privacy risks when sharing private data with untrusted parties [4]. For instance, the individuals own some private photos in social networks which only themself and their friends can access. They want to train a classifier to support automated image annotation. However, they hesitate to share the photos with other entities that they do not trust because it may cause the leakage of personal privacy. Therefore, it is a crucial problem to perform collaborative training while protecting privacy.

To address this issue, many privacy-preserving classifier training schemes have been proposed. The most common cryptographic method for protecting data is homomorphic encryption [5, 6]. However, the homomorphic encryption technique is usually involved with computationally expensive cryptographic primitives, which result in heavy computation cost. Differential privacy is another method to guarantee the security of data [7, 8]. Nevertheless, the differential privacy technique cannot achieve high accuracy because it protects data privacy by adding immeasurable noises to the parameters of the model. Besides, in the above privacy-preserving training schemes, the data owners completely lose control of their data when outsourcing the data to the untrusted parties. The data ownership has not been well guaranteed, which is also a potential threat of data confidentiality. Once the data are shared to the untrusted servers for training, there are potential risks that the data might be modified or replicated by the unauthorized servers. Recently, several privacy-preserving schemes are proposed to achieve privacy-preserving training by utilizing the blockchain technique due to its decentralized digital ledger property. Shen et al. [9] proposed a privacy-preserving SVM training scheme over blockchain-based encrypted IoT data. However, their scheme just fits linear data but cannot deal with classification tasks for nonlinear datasets, which are more common in practice. Moreover, they adopted the computing-intensive Paillier homomorphic encryption technique, which causes huge computation overheads.

In this paper, we propose a new privacy-preserving nonlinear SVM classifier training scheme based on blockchain. Specifically, we establish a blockchain-based data sharing and computation outsourcing mechanism which allows participants to collaboratively train the model while protecting privacy. We utilize the blockchain technique to establish a distributed data sharing environment. In our scheme, the communication of multiple parties is recorded on the blockchain. It ensures transparent delivery of training tasks and the guarantee of data ownership. The blockchain also supports fair incentives to avoid deceptive behavior. We adopt the additive secret sharing technique based on two-party computation to design the computation protocols, which can achieve secure SVM training with minimal computation overheads.

The main contribution of this paper can be summarized as follows:

(1) To train a high-quality nonlinear SVM classifier while protecting privacy, we propose a privacy-preserving training scheme based on blockchain. We utilize the blockchain technique to design a decentralized scheme for collaborative training while ensuring the invariance and ownership of training data. The incentive mechanism of blockchain can also help to guarantee the fairness of the participants and prevent destructive behaviors of the computing parties meanwhile.

(2) We adopt the additive secret sharing techniques and design a series of arithmetic primitives such as multiplication, comparison, and natural exponential computation to realize efficient collaborative training while protecting the privacy of both the data and the model. The protocols in our scheme contain no computation-intensive cryptographic primitives and greatly reduce the computation overheads.

(3) We thoroughly analyze the security of our scheme and conduct experiments over real-world datasets. The security analysis shows that our scheme can well protect the privacy of the data and the users. We conduct comprehensive experiments, and the results demonstrate that our scheme can achieve high accuracy and obtain nonlinear SVM classifiers with practical training efficiency.

The rest of this paper is organized as follows: Section 2 introduces the formulation of the problem, our design goals, and the preliminaries of our scheme. In Section 3, we present the secure computation protocols based on secret sharing. Section 4 gives the details of our privacy-preserving training scheme. Security analysis and performance evaluation are presented in Section 5 and Section 6, respectively. We conclude the paper in Section 8.

2. Problem Statement

2.1. System Model. In this paper, we focus on designing a scheme for privacy-preserving and efficient nonlinear SVM classifier training. There are three entities in our framework: the blockchain, the data providers, and the servers, as shown in Figure 1. The role of each entity is described as follows:

Blockchain. The blockchain in our scheme serves as a distributed and immutable ledger. Each block of the blockchain stores a group of transactions of the training requests, the delivery of training data, and so on. The blockchain also provides an incentive mechanism to guarantee fairness.

Data Providers. The data providers can be institutions, enterprises, or individuals who own some training data and participate in the collaborative training of the SVM classifier. They take charge of encrypting the original datasets before sending them to the servers and generating random values in the secure computation process.

Servers. The servers are two computing parties which are selected from the parties in the decentralized network. They are incented to conduct the training tasks, like the miners in Bitcoin.

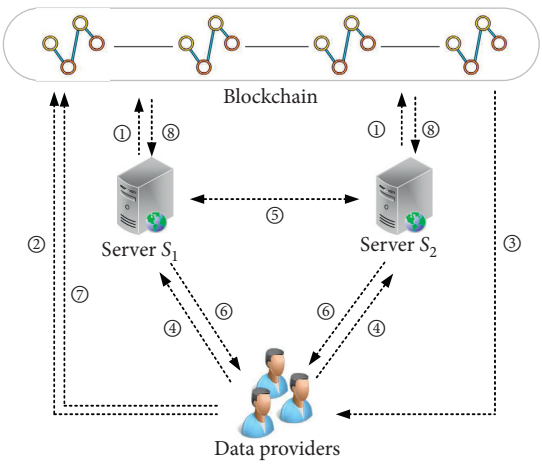

FIGURE 1: System model of the proposed scheme: ① register and $deposit; ② request and $payment; ③ *addr* of servers; ④ encrypted data and random values; ⑤ secure computing protocols; ⑥ encrypted model; ⑦ acknowledgement messages; ⑧ rewards or penalties.

In the blockchain-based decentralized system, the servers should first register on the blockchain and join the network. Meanwhile, each party is required to commit a deposit, which is frozen in the account. The goal of freezing a deposit is to prevent dishonest behavior of the servers. It will be withheld as forfeit if the servers return deceitful results. When the data providers want to conduct collaborative training by utilizing the computing resources in the network, they should generate a transaction that contains the requests and pays. After receiving the transaction, the blockchain selects two servers as computing parties and sends the addresses and the public encryption keys of the servers to the data providers. Then, each provider encrypts the training data with the respective public keys and sends the encrypted data to the servers. On receiving the encrypted data from all the participants, the servers perform the secure computation protocols and conduct the SVM classifier training. The secure computation involves some random values, which can be generated offline by the data providers. After finishing the training process, the servers send the results back to the data providers. Then, each provider reconstructs the model and sends to the blockchain an acknowledgement message, with which the blockchain determines to give the servers rewards or penalties.

2.2. Threat Model. In this paper, we assume that the servers are noncolluding and there are secure channels among the nodes in the decentralized network. We consider the servers to be honest but curious. It means that the servers would execute the designed task honestly, but they are curious to infer sensitive information from the encrypted data and the interactive messages. The privacy threats mainly come from two aspects: (1) the encrypted original datasets. The original datasets may contain lots of sensitive information about the data providers. The adversaries can still infer valuable information about the local data through the training process. (2) The training results. The results of training, i.e., the SVM model, can be used for some commercial benefit. Thus, the results could be embezzled by the computing parties or adversaries.

Specifically, to better evaluate the security of our scheme, we consider two levels of threat as follows:

Level 1. The computing parties can learn nothing about the original datasets. They are assumed to only know the split shares of the datasets and the immediate results of each step in the training protocol.

Level 2. Apart from the information in Level 1, the computing parties can also obtain part of original datasets from certain data providers among the multiple participants of the collaborative training.

2.3. Design Goals. To implement privacy-preserving and efficient nonlinear SVM classifier training, our scheme should meet the following goals:

Security. Our scheme is supposed to protect the information and resist the potential adversities introduced in the threat model. Besides, even if some data providers collude with the servers, the privacy of datasets of other data providers stored on the servers is still preserved.

Correctness. Our training algorithm should be designed correctly to obtain a high-quality SVM classifier model. The privacy-preserving classifier ought to be almost equally effective as those obtained in the plaintext domain, and the accuracy should be high enough for practical use.

Efficiency. High efficiency is the premise for a training algorithm to be used in practical applications. For this purpose, the encryption technique we adopt should not contain computation-intensive cryptographic primitives, and our scheme ought to achieve minimal computing overhead.

2.4. Preliminaries

2.4.1. SVM Classifier. The support vector machine (SVM) is a widely used learning method based on the structural risk minimization [10]. The objective of SVM is to find an optimal separating hyperplane with the maximum margin that distinctly classifies the data points. Suppose that $(\vec{x}_i, y_i)$ is a pair of instance, where $\vec{x}_i$ is a vector containing attributes of the i^{th} instance and y_i is the class label which satisfies $y_i \in \{-1, +1\}$. The decision function to classify an instance $\vec{x}$ is $y = w^T\vec{x} + b$. The optimization problem of the SVM classifier is described as

$$\arg\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i,$$
$$\text{subject to} \quad y_i(w \cdot \vec{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0, \quad \text{for } i = 1, \ldots, m. \tag{1}$$

In practice, the sample data are not linear separable in some classification tasks. That is, there exists no hyperplane

which separates the data points. The nonlinear SVM can map the training data from the low-dimensional space into a higher dimensional space through a kernel function. The corresponding decision function is $y = \sum_{i=1}^{m} \alpha_i^* y_i K(\vec{x}_i, \vec{x}) + b$. The commonly used kernel functions include linear kernel, polynomial kernel, and Gaussian kernel. In this paper, we choose the Gaussian kernel which function is

$$K\left(\vec{x}_i, \vec{x}_j\right) = \exp\left(-\frac{\left\|\vec{x}_i - \vec{x}_j\right\|^2}{2\sigma^2}\right). \quad (2)$$

2.4.2. Blockchain. Blockchain is a kind of distributed ledger technology (DLT) on a peer-to-peer (P2P) network which makes the history of any digital transactions unalterable and transparent through the use of decentralization and cryptographic hashing [11, 12]. The transactions are stored in blocks which are linked as a chain. Blockchain has been originally proposed for constructing a public-distributed ledger for transactions in Bitcoin [13], which is a worldwide electronic payment system.

Blockchain is well known for its decentralization, transparency, and tamper-proof property. It is built with no need for a trusted third party or a central administrator. There is no single-point-of-failure since the blockchain is not maintained by a single party but kept by all parties. The transactions that occur in the network would be recorded on the digital ledger, and no one can modify it. Blockchain usually adopts consensus protocols to manage the right of creating new blocks. There are three kinds of common consensus protocols, i.e., Proof-of-Work (PoW), Proof-of-Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT). Recently, there are lots of attempts of applying blockchain into different scenarios, such as solving trust crisis, simplifying various procedures, and verifying digital identities.

2.4.3. Secure Multiparty Computation. Secure multiparty computation (SMC) originates from the secure two-party computation (2PC) in 1982 (for the so-called Millionaires' Problem) introduced by Yao [14]. SMC is a central cryptographic task that allows multiple parties to jointly compute some function over their inputs while protecting the privacy. Specifically, there are multiple participants $P_1, P_2, \ldots P_n$. Each participant P_i holds its input x_i, and the participants agree on some function f that takes the n inputs. Their goal is to compute $y = f(x_1, x_2, \ldots, x_n)$ while satisfying the following two conditions:

(i) Correctness: the value of y is correctly computed

(ii) Privacy: the participants cannot learn anything about the inputs of others

Most of the SMC architectures are based on some cryptographic tools such as homomorphic encryption (HE), garbled circuits (GC), and oblivious transfer (OT). However, these frameworks are computationally expensive and difficult to be deployed for processing large-scale data sets. By contrast, SMC frameworks based on secret sharing do not involve any cryptographic primitives. Therefore, they could obtain a better performance.

3. Secret Sharing-Based Secure Computing Protocols

In this section, we mainly introduce the secure computation protocols used in our scheme which are based on additive secret sharing. Additive secret sharing is a kind of cryptography technique that means all intermediate values are additively shared between the chosen worker servers. Given two inputs a and b, they will be randomly split into two shares, i.e., $a = a_1 + a_2$ and $b = b_1 + b_2$, which are outsourced to two servers $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Here, we denote $\langle a\rangle_1$ and $\langle a\rangle_2$ as the two shares stored on $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. To cooperatively work out $f(a, b)$, $\mathcal{S}_1$ outputs f_1 and $\mathcal{S}_2$ outputs f_2. f_1 and f_2 satisfy that $f_1 + f_2 = f$. During the computation, $\mathcal{S}_1$ and $\mathcal{S}_2$ learn no information about the value of a, b, and the result f.

3.1. Secure Addition/Subtraction Protocol. Given two values a and b, the addition/subtraction protocol is to jointly compute $a \pm b$. It is obvious that the computation can be executed by $\mathcal{S}_1$ and $\mathcal{S}_2$ independently since $(\langle a\rangle_1 \pm \langle b\rangle_1) + (\langle a\rangle_2 \pm \langle b\rangle_2) = (\langle a\rangle_1 + \langle a\rangle_2) \pm (\langle b\rangle_1 + \langle b\rangle_2) = a \pm b$. Note that there is no interaction between the two servers during the computation.

3.2. Secure Multiplication Protocol. The multiplication protocol is to calculate the product of two given values a and b. We adopt the *Beaver's precomputed multiplication triplets* [15] technique to realize multiplication protocol. The steps of our secure multiplication protocol SecMul$(\cdot)$ are given as follows.

To obtain $c = a \times b$, the algorithm utilizes a pregenerated triplet (u, v, w), where u and v are randomly generated and $w = u \times v$. The shares of u, v, and w are u_i, v_i, and w_i $(i = 1, 2)$, which are stored in $\mathcal{S}_i$, respectively. The servers $\mathcal{S}_i$ then calculate $\langle e\rangle_i = \langle a\rangle_i - \langle u\rangle_i$ and $\langle f\rangle_i = \langle b\rangle_i - \langle v\rangle_i$ locally. After that, they send $\langle e\rangle_i$ and $\langle f\rangle_i$ to each other and reconstruct e and f. Finally, $\mathcal{S}_i$ computes and outputs the shared results as $\langle c\rangle_i = f \cdot \langle a\rangle_i + e \cdot \langle b\rangle_i + \langle w\rangle_i + (i-1) \cdot e \cdot f$.

Thus, the product c can be reconstructed simply by adding the respective results of $\mathcal{S}_1$ and $\mathcal{S}_2$ as $c = \langle c\rangle_1 + \langle c\rangle_2$.

3.3. Secure Comparison Protocol. Given a value a and b, the secure comparison protocol SecComp$(\cdot)$ is used to judge whether $a < b$. Specifically, the function outputs 1 if and only if $a < b$ and outputs 0 otherwise. We adopt the bit-decomposition method in [16] and follow the comparison protocol proposed by Huang et al. [17] which is based on additively secret sharing.

We first transform the real-number shares into integers. Specifically, we multiply the numbers by 10^p and truncate the remaining decimal parts. Then, we utilize the two's complement representation, where the most significant bit (MSB) of a number indicates whether it is a positive or

negative. For an l-bit signed number c, its binary complement form can be denoted as $c^{(l-1)}, c^{(l-2)}, \ldots, c^{(0)}$. Correspondingly, c can be reconstructed as

$$c = -c^{l-1} \cdot 2^{l-1} + \sum_{j=0}^{l-2} c^{(j)} \cdot 2^j. \quad (3)$$

Suppose that the l-bit shares of c are c_1 and c_2, and the protocol performs bitwise operations over c_1 and c_2 to compute the sign of c. Thus, the protocol can compare a and b by computing the sign of $a - b$.

3.4. Secure Natural Exponential Protocol. Given a value a, the secure natural exponential protocol $\mathrm{SecExp}(\cdot)$ is used to calculate e^a. The algorithm utilizes $\mathrm{SecMul}(\cdot)$ and the property of additive secret sharing. The shares of a are $\langle a \rangle_1$ and $\langle a \rangle_2$. $\mathcal{S}_1$ and $\mathcal{S}_2$ calculate $e^{\langle a \rangle_1}$ and $e^{\langle a \rangle_2}$ on the local servers, respectively. Note that $\langle a \rangle_1 + \langle a \rangle_2 = a$, and we can calculate e^a as

$$e^{\langle a \rangle_1} \cdot e^{\langle a \rangle_2} = e^{\langle a \rangle_1 + \langle a \rangle_2} = e^a. \quad (4)$$

Thus, $\mathcal{S}_1$ randomly splits $e^{\langle a \rangle_1}$ into two parts, i.e., $\langle e^{\langle a \rangle_1} \rangle_1$ and $\langle e^{\langle a \rangle_1} \rangle_2$, and sends $\langle e^{\langle a \rangle_1} \rangle_2$ to $\mathcal{S}_2$. Correspondingly, $\mathcal{S}_2$ randomly splits $e^{\langle a \rangle_2}$ into $\langle e^{\langle a \rangle_2} \rangle_1$ and $\langle e^{\langle a \rangle_2} \rangle_2$ and sends $\langle e^{\langle a \rangle_2} \rangle_1$ to $\mathcal{S}_1$. Finally, $\mathcal{S}_1$ and $\mathcal{S}_2$ conduct $\mathrm{SecMul}(\cdot)$ to calculate $e^{\langle a \rangle_1} \cdot e^{\langle a \rangle_2}$.

4. The Proposed Scheme

In this section, we present the framework of our privacy-preserving nonlinear SVM classifier training scheme. Our scheme contains two main parts: the blockchain design and the privacy-preserving SVM classifier training. The details are as follows.

4.1. Blockchain Design

4.1.1. Registering. The parties that want to join the decentralized network and become computing parties should first create a registering transaction in the blockchain. Each party would send a register request to the blockchain which is supposed to own at least \$ deposit in the account, which is to be frozen by the contract when registering. It is used to avoid malicious behaviors during the computing period. Specifically, the servers may reduce the quantity of training data to save computation resources. They may also forge inaccurate results and return them to the data providers.

4.1.2. Consensus Protocol. In the decentralized network, a consensus protocol is necessary to make all the nodes in the network reach a consensus. We adopt the Proof-of-Work (PoW) protocol as the consensus protocol in our scheme. The computing nodes increment a nonce in the block and compute the hash value of the block header. The first and the second nodes that find such a value that meets the predefined requirement by the contract would broadcast their results in the blockchain. After the other nodes verify the correctness of the results, the two nodes are accepted as the computing parties, and the first node has the right to create a new block.

4.1.3. Computing Request. The data provider who wants to outsource computation tasks to the computing nodes should first generate a transaction as $\mathrm{Tran}_{\mathrm{req}} = (\mathrm{protocols}, \$\,\mathrm{payment})$, in which protocols are the computing functions for the servers to execute and \$ pay is the payment. Then, the data owner publishes the request transaction on the blockchain.

After receiving the request from the data provider, the blockchain selects two servers for computing from the nodes in the network by PoW protocol and publishes the addresses $\mathrm{addr}_i\,(i = 1, 2)$ and the public key $\mathrm{pk}_i\,(i = 1, 2)$ of the two computing nodes.

4.1.4. Payment. If there are multiple data providers who own a set of training data and want to perform collaborative training, they should first reach a consensus about their payments of the training. In a practical scenario, each data provider usually owns different amounts of training data. The payments ought to be decided based on the amounts of data that the providers contribute. Specifically, the data provider who contributes a larger amount of training data ought to give less payment for the collaborative training. Once the data providers receive the results and believe the results can meet their requirements, they would send an acknowledgement message to the blockchain. If the blockchain receives more than two-thirds of the acknowledgement messages from data providers, it splits the total payment and distributes the payments to the two computing servers, respectively. Otherwise, the blockchain would return the payments to the data providers and deduct the fine from the deposits of the computing servers. The criterion of rewards and punishments can be adjusted in the consensus protocol before training.

4.2. Privacy-Preserving SVM Classifier Training

4.2.1. System Initialization. Suppose that there are n data providers $\mathrm{DP}_j\,(1 \le j \le n)$ who own some training data, respectively, and want to collaboratively train an SVM classifier with a kernel function. The data providers first reach a consensus on the training protocols, parameters, and payments. Then, they send the transactions of request to the blockchain and thereafter receive the addresses and public keys of the computing servers.

After that, the data providers first encrypt the training data by randomly splitting each element into two shares. Then, they encrypt the shares with the corresponding public key pk_i of the two computing servers and obtain the encrypted training datasets $\langle D_j \rangle_i$. Finally, the data providers send the encrypted datasets to the two servers, respectively, and then publish the transactions on the blockchain.

4.2.2. Training. After receiving all the encrypted datasets from the data providers, the servers decrypt the shares by utilizing their corresponding private key sk_i and perform the

training protocol. In our SVM model, we choose the Gaussian kernel function, which is depicted in equation (2), to achieve nonlinear separation. The function can be interactively calculated by the two servers. The steps of calculating the Gaussian kernel are shown in Algorithm 1.

To train the SVM classifier, we adopt the gradient descent (GD) as the optimization method, which is utilized in [9]. Compared with another optimization algorithm that is also frequently used in plaintext tasks, i.e., the sequential minimal optimization (SMO), GD contains less complex computation. Thus, it is regarded to be more suitable for the training in the encrypted domain. By introducing a hinge loss, the optimization problem of the SVM is converted to

$$\min \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \max\left(0, 1 - y_i\left(\sum_{j=1}^{m} \alpha_j K\left(\vec{x}_i, \vec{x}_j\right) + b\right)\right). \tag{5}$$

The protocol firsts executes SecComp$(\cdot)$ to compare $y_i(\sum_{i=1}^{m} \alpha_i K(\vec{x}_i, \vec{x}) + b)$ and 1. If $y_i(\sum_{i=1}^{m} \alpha_i K(\vec{x}_i, \vec{x}) + b) < 1$, the servers update α and b by calculating the derivatives of the margin and the hinge loss. The steps of privacy-preserving training are shown in Algorithm 2. The dataset for training and the SVM model is well protected during the training process. The servers and other adversaries cannot infer any information except the respective shares obtained in each step.

5. Security Analysis

In this section, we present the security strength of our proposed scheme under the two-level threat models. First, we analyze the security of our scheme under the Level 1 threat model based on the universal composability (UC) framework [18], which is regarded to guarantee strong security properties. To prove the security of our scheme, we first give some definitions as follows.

Definition 1. A protocol is secure if there exists a probabilistic polynomial-time simulator $\mathcal{S}$ that can generate a view for the adversary $\mathcal{A}$ in the real world which is computationally indistinguishable from its real view.

Due to the secret sharing-based protocols, the addition and subtraction operations which are computed locally on the servers can be easily simulated. We prove the security of other computing protocols in our scheme.

Theorem 1. *The protocol SecMul$(\cdot)$ is secure under the honest but curious model.*

Proof. The view of $\mathcal{S}_1$ is $\text{view}_1 = (a_1, b_1, u_1, v_1, w_1, e_2, f_2)$. It is obvious that a_1 and b_1 are randomly split from a and b and u_1, v_1, and w_1 are uniformly random values. e_2 and f_2 are also random values because they are generated as $e_2 = a_2 - u_2$ and $f_2 = b_2 - v_2$. The output of $\mathcal{S}_1$ is $\text{view}_1 = f \cdot a_1 + e \cdot b_1 + w_1$, which is also uniformly random. Note that both the input and output of $\mathcal{S}_1$ are random values, so they can be perfectly simulated by $\mathcal{S}$. The view of adversary $\mathcal{A}$ and its real view are computationally indistinguishable. Similarly, the input and output of $\mathcal{S}_2$ can also be perfectly simulated.

Theorem 2. *The protocol SecComp$(\cdot)$ is secure under the honest but curious model.*

Proof. For the comparison protocol SecComp$(\cdot)$, the view_1 and view_2 of $\mathcal{S}_1$ and $\mathcal{S}_2$ are $\text{view}_1 = (a_1, u_1, v_2)$ and $\text{view}_2 = (a_2, u_2, v_1, v_2^{(l-1)}, v_2^{(l-2)}, \ldots, v_2^{(0)})$. The values are random and simulatable. The bitwise addition can be deployed by secure addition and secure multiplication, which has been proved to be simulatable. Therefore, it can be proved that the comparison protocol can be simulated by a simulator $\mathcal{S}$.

Theorem 3. *The protocol SecExp$(\cdot)$ is secure under the honest but curious model.*

Proof. The SecExp$(\cdot)$ protocol in our scheme only involves natural exponential computation, subtraction, and secure multiplication. The first two operations are implemented locally on the servers. The secure multiplication is proved that it can be simulated. Thus, the secure natural exponential computation is simulatable. A view can be generated for the adversary $\mathcal{A}$, and the view is computationally indistinguishable with its real view.

The privacy-preserving training protocol is composed of the above computing protocols, which are proved to be secure. Thus, our privacy-preserving nonlinear SVM classifier training scheme is secure against the Level 1 threat. In the Level 2 threat model, the servers can obtain some original training data from certain data providers. However, the shares of other training data are still randomly generated. Thus, for the datasets from other data providers, the simulator $\mathcal{S}$ can still generate the view that is computationally indistinguishable from its real view. Therefore, the security of the other training data and the protocols can still be guaranteed under the Level 2 threat model.

6. Performance Evaluation

In this section, we evaluate the performance of our scheme by conducting experiments over real-word datasets. We use a real-world dataset about social network advertising collected from a trusted website. The dataset contains 400 instances. Each instance is with 4 features and labeled as purchased or not purchased. We also use the Breast Cancer Wisconsin Database (BCWD) from the UCI machine learning repository. The dataset contains 699 instances, and each instance contains nine features. The instances in the dataset are labeled as benign or malignant. We implement the experiments on a PC with a 32-core Intel i7 CPU @ 1.80 GHz and 16 GB RAM. The algorithms are programmed with Python 2.7. Specifically, we investigate the performance through accuracy and efficiency.

Input: $\mathcal{S}_1$: the shared vectors $\langle \vec{x}_a \rangle_1$, $\langle \vec{x}_b \rangle_1$, and σ; $\mathcal{S}_2$: the shared vectors $\langle \vec{x}_a \rangle_2$, $\langle \vec{x}_b \rangle_2$, and σ.
Output: $\mathcal{S}_1$: the shared Gaussian kernel result $\langle r \rangle_1$; $\mathcal{S}_2$: the shared Gaussian kernel result $\langle r \rangle_2$.
(1) $\mathcal{S}_i$ initialize $\langle s \rangle_i = 0$.
(2) **for** k from 1 to $len(\langle \vec{x}_a \rangle_i)$ **do**
(3) $\mathcal{S}_i$ locally compute $\langle z \rangle_i \longleftarrow \langle x_a[k] \rangle_i - \langle x_b[k] \rangle_i$.
(4) $\mathcal{S}_i$ compute $\langle g \rangle_i \longleftarrow \text{SecMul}(\langle z \rangle_i, \langle z \rangle_i)$.
(5) $\mathcal{S}_i$ locally compute $\langle s \rangle_i \longleftarrow \langle s \rangle_i + \langle g \rangle_i$.
(6) **end for**
(7) $\mathcal{S}_i$ locally compute $\langle f \rangle_i \longleftarrow -(1/2\sigma^2) \cdot \langle s \rangle_i$.
(8) $\mathcal{S}_i$ compute $\langle r \rangle_i \leftarrow \text{SecExp}(\langle f \rangle_i)$.

Algorithm 1: Secure Gaussian kernel function.

Input: $\mathcal{S}_i$: the split dataset $\langle D \rangle_i = \{(\langle \vec{x}_1 \rangle_i, \langle y_1 \rangle_i), (\langle \vec{x}_2 \rangle_i, \langle y_2 \rangle_i), \ldots, (\langle \vec{x}_m \rangle_i, \langle y_m \rangle_i)\}$ learning rate λ, max iterations T, and precision ε.
Output: $\mathcal{S}_i$: $\langle \alpha^* \rangle_i$, $\langle b^* \rangle_i$.
(1) $\mathcal{S}_i$ initialize $\langle \alpha^1 \rangle_i, \langle b^1 \rangle_i, \langle \text{loss} \rangle_i$.
(2) **for** p from 1 to m **do**
(3) **for** q from 1 to m **do**
(4) $\mathcal{S}_i$ compute $\langle K[p][q] \rangle_i \longleftarrow \text{SecKer}(\langle \vec{x}_p \rangle_i, \langle \vec{x}_q \rangle_i)$.
(5) **end for**
(6) **end for**
(7) **While** loss $> \varepsilon$ or $t < T$ **do**
(8) $\mathcal{S}_i$ initialize Δ_α, Δ_b.
(9) $\mathcal{S}_i$ compute $\langle \text{loss} \rangle_i \longleftarrow \text{SecMul}(\text{SecMul}(\langle \alpha^t \rangle_i, \langle K \rangle_i), \langle \alpha^t \rangle_i^T)$.
(10) **for** p from 1 to m **do**
(11) **for** q from 1 to m **do**
(12) $\mathcal{S}_i$ compute $\langle g^p \rangle_i \longleftarrow \text{SecMul}(\langle \alpha^t[q] \rangle_i, \langle K[p][q] \rangle_i)$.
(13) $\mathcal{S}_i$ locally compute $\langle s^p \rangle_i \longleftarrow \langle s^p \rangle_i + \langle g^p \rangle_i$.
(14) **end for**
(15) $\mathcal{S}_i$ locally compute $\langle f^p \rangle_i \longleftarrow \langle s^p \rangle_i + \langle b^p \rangle_i$.
(16) $\mathcal{S}_i$ compute $\langle f^p \rangle_i \longleftarrow \text{SecMul}(\langle y_p \rangle_i, \langle f^p \rangle_i)$.
(17) $\mathcal{S}_i$ compute $\text{SecComp}(\langle f^p \rangle_i, 1)$.
(18) **if** $\langle f^p \rangle_i < 1$ **then**
(19) $\mathcal{S}_i$ compute $\langle \Delta_\alpha \rangle_i \longleftarrow \langle \Delta_\alpha \rangle_i - C \cdot \text{SecMul}(\langle y_p \rangle_i \cdot K[p])$.
(20) $\mathcal{S}_i$ compute $\langle \Delta_b \rangle_i \longleftarrow \langle \Delta_b \rangle_i - C \cdot \langle y_p \rangle_i$.
(21) **end if**
(22) **end for**
(23) $\mathcal{S}_i$ update $\langle \alpha^t \rangle_i \longleftarrow \langle \alpha^t \rangle_i - \lambda \cdot \langle \Delta_\alpha \rangle_i$.
(24) $\mathcal{S}_i$ update $\langle b^t \rangle_i \longleftarrow \langle b^t \rangle_i - \lambda \cdot \langle \Delta_b \rangle_i$.
(25) $\mathcal{S}_i$ compute $\text{SecComp}(\langle \text{loss} \rangle_i, \varepsilon)$.
(26) $t = t + 1$.
(27) **end while**
(28) $\mathcal{S}_i$ return $\langle \alpha^* \rangle_i, \langle b^* \rangle_i$.

Algorithm 2: Secure nonlinear SVM classifier training.

6.1. Accuracy. The precision rate and the recall rate are two key parameters to evaluate the accuracy of a classifier. We calculate the two parameters by utilizing both our proposed privacy-preserving SVM classifier and the traditional SVM classifier over plaintext. The results are shown in Table 1. We can see that our scheme can achieve nearly the same accuracy with the SVM classifier over plaintext. It demonstrates that the cryptographic methods in our scheme do not influence the classification functionality. Our scheme can maintain high accuracy while protecting privacy.

Table 1: Accuracy performance.

	Our scheme (%)	SVM over plaintext (%)
Precision	85.8	86.7
Recall	92.8	91.3

6.2. Efficiency. In this section, we evaluate the efficiency of our scheme. Specifically, we investigate the time consumption both on the data providers and the servers by utilizing cross-validation. We evaluate the time consumption with different percentage of instances for training and

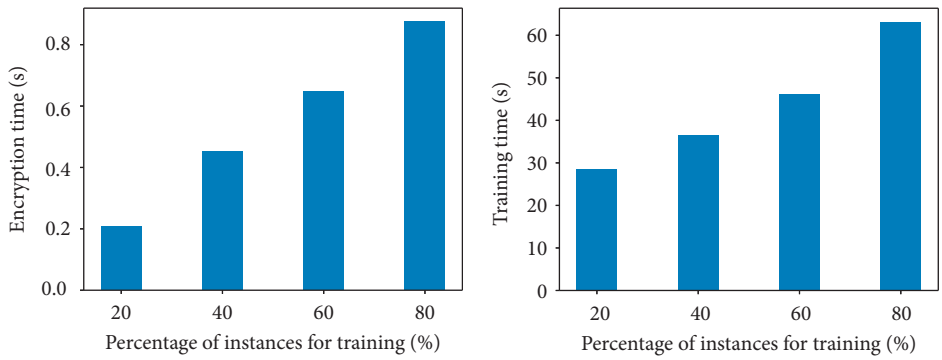

Figure 2: Efficiency performance with different percentage of instances.

testing, as shown in Figure 2. Specifically, we take several percentages of instances for training and the others for testing. We can see that the time consumption of training data encryption is positively correlated with the percentage of instances for training because more instances mean a larger number of computation both on the data provider side and the server side. Overall, the efficiency of our scheme is acceptable for practical use.

6.3. Comparison. We also compare our scheme with the privacy-preserving SVM classifier training scheme proposed by Shen et al. We use the BCWD dataset and conduct training both on our scheme and Shen et al.'s scheme. The results are shown in Table 2. We can see that our scheme can achieve higher accuracy than Shen's scheme. It is because our scheme adopts the nonlinear kernel, which makes the scheme more adaptive to the datasets that contain nonlinear data. As for the efficiency, it is shown in Table 3 that our scheme achieves much better efficiency performance. It is because our scheme does not involve any computationally expensive cryptography techniques. We can see that the time consumption on the data provider side in our scheme is just 1.4 s, while in Shen et al.'s scheme, it takes the data provider more than 2000 s. The time of computation on the server side is also much less than Shen et al.'s scheme. The experiment results show that our scheme is much more efficient than Shen et al.'s scheme and achieves better overall performance for practical utilization.

Table 2: Comparison of the accuracy.

	Our scheme	Shen's scheme (%)
Precision	93.3%	90.35
Recall	1	96.19

Table 3: Comparison of the efficiency.

	Our scheme (s)	Shen's scheme (s)
Server side	146	953
Data provider side	2.6	2233

7. Related Work

Classification is a fundamental task of machine learning and applied to many fields, such as face recognition, speech recognition [19], and financial prediction. To protect the privacy of individuals and enterprises, there have been many privacy-preserving classifier training schemes proposed. These schemes focus on training classifiers and performing classification tasks over encrypted data while protecting user privacy. Bos et al. [20] proposed a scheme to privately conduct medical predictive analysis tasks on encrypted data. However, their scheme cannot protect the model from being exposed to the users. In addition, the scheme leaks much information of the patients. Raphael Bost et al. [5] proposed privacy-preserving protocols for three common classifiers, i.e., hyperplane decision, Naïve Bayes, and decision trees. González-Serrano et al. [21] proposed a privacy-preserving semiparametric SVM scheme by utilizing a partial homomorphic cryptosystem. Recently, Shen et al. [9] proposed a privacy-preserving SVM training scheme based on the blockchain for secure IoT data sharing. They utilized the Paillier encryption technique to design secure building blocks and achieve secure SVM training. However, these schemes all adopt the homomorphic encryption and contain computationally expensive cryptographic primitives, which make the schemes inefficient.

Some existing schemes are based on a combination of homomorphic encryption and multiparty computation. Barni et al. [22] proposed a privacy-preserving neural network classification scheme. In their algorithm, the neural networks contain a series of scalar products which are encrypted by utilizing the homomorphic encryption. The activation functions are calculated based on the secure multiparty computation. Subsequently, Orlandi et al. [23] enhanced the scheme of Barni. They masked the scalar product results and protected the intermediate results from revealing to the client. However, these schemes also suffered heavy computation overheads.

There are also a number of privacy-preserving training schemes that adopt differential privacy. Pathak and Raj [24] presented a scheme to learn a discriminatively trained multiclass Gaussian mixture model-based classifier that preserves differential privacy using a large margin loss

function. Zhang et al. [25] designed a privacy-preserving decision tree classification construction model based on a differential privacy-protection mechanism. Nevertheless, in these schemes, there have been conflicts between privacy and accuracy. The differential privacy-protection technology by adding immeasurable noises influences the accuracy of the models.

8. Conclusion

In this paper, we proposed a new privacy-preserving nonlinear SVM classifier training scheme for social networks. We utilize the blockchain technique to design a decentralized framework for data sharing and training while ensuring the invariance of datasets. We adopt additive secret sharing based on secure two-party computation and design a suite of secure computing protocols to conduct the training process with no information leakage. Our training scheme is proved to be secure through comprehensive analysis. Experiments over real datasets demonstrate that our scheme can achieve high accuracy and efficiency for practical applications.

Data Availability

The data used to support the findings of this study are available at https://github.com/JIANAN17/privacy-preserving-SVM.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work was supported by the National Natural Science Foundation of China under Grant nos. 61672195 and 61872372, the Open Foundation of State Key Laboratory of Cryptology (no. MMKFKT201617), and the National University of Defense Technology (Grant no. ZK19-38).

References

[1] J. Hua, H. Zhu, F. Wang et al., "CINEMA: efficient and privacy-preserving online medical primary diagnosis with skyline query," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1450–1461, 2018.

[2] Z. Ma, Y. Liu, X. Liu, J. Ma, and K. Ren, "Lightweight privacy-preserving ensemble classification for face recognition," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5778–5790, 2019.

[3] M. Ghazanfar and A. Prugel-Bennett, "An improved switching hybrid recommender system using naive bayes classifier and collaborative filtering," in *Proceedings of the International Multiconference of Engineers and Computer Scientists 2010*, Hong Kong, China, March 2010.

[4] X. Yang, Z. Liu, and Li Jin, *Security and Privacy in Social Networks and Big Data*, Springer, Berlin, Germany, 2020.

[5] R. Bost, R. Ada Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proceedings of the NDSS Symposium 2015*, p. 4325, San Diego, CA, USA, February 2015.

[6] H. Zhu, X. Liu, R. Lu, and H. Li, "Efficient and privacy-preserving online medical prediagnosis framework using nonlinear SVM," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 3, pp. 838–850, 2016.

[7] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research: JMLR*, vol. 12, pp. 1069–1109, 2011.

[8] M. Abadi, A. Chu, I. Goodfellow et al., "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, Vienna, Austria, October 2016.

[9] M. Shen, X. Tang, L. Zhu, X. Du, and M. Guizani, "Privacy-preserving support vector machine training over blockchain-based encrypted iot data in smart cities," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7702–7712, 2019.

[10] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.

[11] T. Aste, P. Tasca, and T. Di Matteo, "Blockchain technologies: the foreseeable impact on society and industry," *Computer*, vol. 50, no. 9, pp. 18–28, 2017.

[12] H. Huang, X. Chen, Q. Wu, X. Huang, and J. Shen, "Bitcoin-based fair payments for outsourcing computations of fog devices," *Future Generation Computer Systems*, vol. 78, pp. 850–858, 2018.

[13] M. Crosby, Nachiappan, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: beyond bitcoin," *Applied Innovation Review*, vol. 2, no. 6–10, p. 71, 2016.

[14] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pp. 160–164, Chicago, IL , USA, November 1982.

[15] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Proceedings of the Annual International Cryptology Conference*, pp. 420–432, Santa Barbara, CA, USA, August 1991.

[16] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation," in *Proceedings of the Theory of Cryptography Conference TCC 2006*, pp. 285–304, New York, NY, USA, March 2006.

[17] K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, "A lightweight privacy-preserving CNN feature extraction framework for mobile sensing," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[18] C. Ran, A. Cohen, and Y. Lindell, "A simpler variant of universally composable security for standard multiparty computation," in *Proceedings of the Annual Cryptology Conference*, pp. 3–22, Santa Barbara, CA, USA, August 2015.

[19] Z. Liu, Z. Wu, T. Li, J. Li, and C. Shen, "GMM and CNN hybrid method for short utterance speaker recognition," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3244–3252, 2018.

[20] J. W. Bos, K. Lauter, and M. Naehrig, "Private predictive analysis on encrypted medical data," *Journal of Biomedical Informatics*, vol. 50, pp. 234–243, 2014.

[21] F.-J. González-Serrano, Á. Navia-Vázquez, and A. Amor-Martín, "Training support vector machines with privacy-protected data," *Pattern Recognition*, vol. 72, pp. 93–107, 2017.

[22] M. Barni, C. Orlandi, and A. Piva, "A privacy-preserving protocol for neural-network-based computation," in *Proceedings of the 8th Workshop on Multimedia and Security, MM&Sec 2006*, pp. 146–151, Geneva, Switzerland, September 2006.

[23] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing via homomorphic encryption," *EURASIP Journal on Information Security*, vol. 2007, no. 1, pp. 1–11, 2007.

[24] M. A. Pathak and B. Raj, "Large margin Gaussian mixture models with differential privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 4, pp. 463–469, 2012.

[25] L. Zhang, Y. Liu, Y. Liu, R. Wang, X. Fu, and Q. Lin, "Efficient privacy-preserving classification construction model with differential privacy technology," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 170–178, 2017.