Secure SVM Training Over Vertically-Partitioned Datasets Using Consortium Blockchain for Vehicular Social Networks

Meng Shen, *Member, IEEE*, Jie Zhang, Liehuang Zhu, *Member, IEEE*, Ke Xu, *Senior Member, IEEE*, and Xiangyun Tang

***Abstract*—Machine learning (ML) techniques are expected to be used for specific applications in Vehicular Social Networks (VSNs). Support vector machine (SVM) is one of the typical ML methods and widely used for its high efficiency. Due to the limitation of data sources, the data collected by different entities usually contain attributes that are quite different. However, in some real-world scenarios, when training an SVM classifier, many entities face the same problem that they are lacking in data with adequate attributes. Thus multiple entities are required to share data to combine a dataset with diverse attributes and then jointly train a comprehensive classifier. However, data privacy concerns are raised because of data sharing. To sovle the problem, we propose a privacy-preserving SVM classifier training scheme over vertically-partitioned datasets posessed by multiple data providers. In our scheme, we utilize consortium blockchain and threshold homomorphic cryptosystem to establish a secure SVM classifier training platform without a trusted third-party. We keep lots of training operations locally over original data and necessary interactions between participants are protected by the threshold Paillier and consortium blockchain. Security analysis proves that our scheme can preserve the privacy of the original data and the training intermediate values. Extensive experiments indicate that our scheme has high efficiency and no accuracy loss.**

***Index Terms*—Privacy Preserving, Vehicular Social Networks, support vector machine, consortium blockchain.**

I. Introduction

NOWADAYS, the development of cloud computing and edge computing has led to a surge in the amount of data generated in different scenario including Vehicular Social Networks (VSNs) [1]–[6]. Efficient methods are in urgent need to optimize the performance of VSNs from the aspects of safety, amenity, convenience and entertainment [4]. Machine learning (ML) and deep learning have drawn the attention of reseachers from many domains [7]–[11]. Thus, related technologies especially machine learning and deep learning are gradually applied to VSNs data analysis. Among the frequently-applied methods, with its efficient performance and high robustness, support vector machine (SVM) has a wide range of applications in many scenarios. For example, SVM is employed to detect negative communication conditions [1].

In the field of VSNs, data are collected by different entities [12], [13] such as vehicular manufacturers, vehicle management agencies, and social network application companies. Data held by those entities usually differ in attributes due to the different data sources. Because of the limitation of data sources, a single entity in VSNs seldom has a training dataset with comprehensive attributes. For example, most of the attributes about vehicle locations and mobility traces are held by vehicle network service platforms, and user preference attributes may be mainly owned by social network application companies. It is obvious that the performance of an SVM classifier is determined by the dataset. Thus, when training an SVM classifier for VSNs applications, entities share data to aggregate a dataset with sufficient attributes, before obtaining a more efficient classifier. From another perspective, the dataset owned by each entity can be treated as a vertically-partitioned part of the merged dataset.

However, there exist several serious security challenges when entities share data to train an SVM classifier. On one hand, since VSNs data contains much private information (e.g., vehicle location, user preference), data sharing is limited by the constantly enacted regulations, which restrict companies from using and sharing user's private information worldwide. On the other hand, VSNs data contains rich information with high value, making data providers reluctant to share their original data directly. Without a suitable privacy-preserving mechanism, data are captured by other entities, leading to value loss of shared data for data owners. Security challenges in intelligent and connected vehicles have been researched [14], and data sharing between entities should also be well studied.

For a long time, privacy disclosure isuue raised in diverse scenarios has been highly concerned [15]–[26]. Among those scenarios, many researches pay attention to train a machine learning classifier securely over both horizontally and vertically

Manuscript received September 12, 2019; revised November 3, 2019; accepted November 17, 2019. Date of publication December 3, 2019; date of current version June 18, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0803405, in part by the National Natural Science Foundation of China under Grants 61972039, 61872041, and 61602039, in part by the Beijing Natural Science Foundation under Grant 4192050, in part by the China National Funds for Distinguished Young Scientists under Grant 61825204, and in part the Beijing Outstanding Young Scientist Program under Grant BJJWZYJH01201910003011. The review of this article was coordinated by Prof. H. Li. *(Corresponding author: Liehuang Zhu.)*

M. Shen, J. Zhang, L. Zhu, and X. Tang are with the School of Computer Science, Beijing Institute of Technology and State Key Laboratory of Cryptology, Beijing 5159, China (e-mail: shenmeng@bit.edu.cn; 3120181068@bit.edu.cn; liehuangz@bit.edu.cn; tangguotxy@163.com).

K. Xu is with the Department of Computer Science, Tsinghua University and Beijing National Research Center for Information Science and Technology (BNRist), Beijing 100084, China (e-mail: xuke@mail.tsinghua.edu.cn).

Digital Object Identifier 10.1109/TVT.2019.2957425

partitioned datasets. Many existing solutions adopt secure multi-party computation (SMC) to prevent privacy disclosure. Firstly, in those schemes, how to balance security and efficiency issues still faces big challenges. Then, one or more aided servers are essential with the assumption that they are trusted or semi-trusted during the training process. Obviously, in a real-world scenario, it is impractical to provide such aided servers for the participants. To deal with the two challenges of applying the privacy protection scheme to real-world scenarios, we propose an efficient and secure SVM classifier training scheme based on consortium blockchain where no third party is introduced.

Firstly, our scheme addresses the first challenge of how to preserve privacy in an effective way. Since differential privacy is unable to guarantee the high-level security and may get an incorrect classifier as well, threshold homomorphic cryptosystem is exploited in our scheme [27]. Considering there are many time-consuming operations in the process of encryption and decryption of homomorphic cryptosystem, to bridge the gap between efficiency and security, in our scheme, each participant keeps most calculations during the SVM training process locally with original data, and necessary intermediate values are encrypted to be shared with other entities. To further reduce the sharing frequency of the intermediate values, we use gradient descent as the optimization algorithm. At each iteration, each participant only needs to share their intermediate values for one time which means one participant needs to encrypt and decrypt for only one time.

The second challenge is how to preserve the data privacy without introducing a trusted-third party (aided server) during the training process. We apply the threshold Paillier to avoid introducing a trusted third-party and both encryption and decryption operations can be done among those participants [28]. In the threshold Paillier, the private key is divided into several sub-private keys and the participants whose number exceed the threshold can decrypt the encrypted messages. At the same time, relying on consortium blockchain, our scheme builds up a secure and public data sharing platform which provides secure communication for participants when they share intermediate values. By this way, in one interaction, a participant can share data with others or get all the other participants' shared data instead of building and keeping several peer to peer secure communication channels between any two participants. Last but not least, consortium blockchain's authority management schemes and access control mechanisms protect data from being obtained by entities outside the participants [29], [30].

The main contributions of this paper are as follows:

1) Based on blockchain techniques, we propose an efficient and secure SVM training scheme between several VSNs data providers whose data have different attributes. Our scheme is capable of providing an open, reliable and transparent decentralized data sharing platform, and it runs without a trusted third-party.
2) With the introduction of threshold homomorphic cryptosystem, we establish a privacy-preserving intermediate value sharing platform in a decentralized condition. In our scheme, most training calculations are performed locally by VSNs data providers using their original data, and thus much encryption calculation cost is avoided.
3) Our scheme can tolerate arbitrary subset of participants to collude with noncritical information disclosure. Meanwhile, extensive experiments are conducted on VSNs datasets and the results demonstrate that the classifier trained by our scheme has no accuracy loss compared with the classifier trained in general conditions.

The remainder of this paper is organized as follows. In Section II, we sort out the related work and review the preliminaries in Section III. In Section IV, we describe the problem to be solved, before giving the threat model and design goal. Then, we introduce our scheme in detail in Section V. After that, security analysis is conducted in Section VI and experiments are carried out to evaluate our scheme in Section VII. At last, we present a short conclusion in Section VIII.

II. Related Work

Existing researches on privacy preserving machine learning are involved in many ML methods including traditional methods such as linear regression, SVM, naive bayes classifier, and logic regression. Deep learning [31] is also focused in the last few years.

Gascon *et al.* [32] train a linear regression classifier with vertically partitioned datasets by a hybrid protocol. In this protocol, garbled circuits are used in the two-party computation. A crypto service provider is needed in two-party cases, and a crypto service provider and an evaluator is needed at the same time in multi-party case.

Nikolaenko *et al.* [24] design a privacy-preserving ridge regression algorithm which can be divided into two phases and each phase uses homomorphic encryption and Yao garbled circuits separately. In the algorithm, an evaluator and a crypto service provider are essential to realize the algorithm.

Mohassel *et al.* [17] present a protocol for privacy preserving machine learning and this protocol is able to support several machine learning algorithms such as linear regression, logistic regression and neural network. In this scheme, two servers collect data from data providers and train a model in a secure way by two-party computation. At the same time, the two servers cannot collude.

Mohassel *et al.* [19] construct a framework where three servers are necessary to train linear regression, logistic regression and neural network models based on three-party computation.

Abadi *et al.* [33] use differential privacy to protect sensitive information in datasets during deep learning. Although differential privacy is an efficient method of privacy protection, the introduction of perturbations has a negative impact on the accuracy of the final trained classifier. In contrast, homomorphic encryption is a more accurate privacy protection scheme.

Since homomorphic encryption is computationally intensive, some homomorphic encryption operations only support either addition or multiplication. In some schemes using homomorphic encryption, a trusted third party is introduced. Francisco-Javier *et al.* [34] use a two-server model based on partial homomorphic encryption to solve the privacy protection problem when multiple data providers train SVM models.

TABLE I
NOTATIONS

Notations	Description
D	The dataset combined by several VDPs
D^A	The dataset of VDP A
d	The dimension of dataset D
d^A	The dimension of dataset D^A
x_i^A	The i-th data instance of dataset D
y_i	The i-th data label
λ	Learning rate
Δ_t	Gradient
m	The size of dataset D
$[[M]]$	The encryption of m under Paillier
w, b	The model parameters

III. PRELIMINARIES

A. Notation

Consider dataset D is combined with several VDPs who have its own dataset D^p $p \in A, B, \ldots N$, where x_i^p represents the i-th instance in D^p, and y_i is shared as a data label between all the related i-th instance x_i^X. When training an SVM classifier, we define w as the model parameter, Δ_t as gradient in the t iteration, and λ as the learning rate. Meanwhile, we assume $[[M]]$ as the encryption of message M under Paillier. Table I shows the notations used in this paper.

B. Classifier of SVM

SVM also known as support vector machine, is a two-category model [35]. The main idea is to find a hyperplane in the instance space and the hyperplane can classify different types of test instances. The hyperplane can be expressed as $y = w^T x + b$, $(x_i, y_i) \in D$. If $y = w^T x_i + b \geq 1, y_i = +1$; Else if $y = w^T x_i + b \leq -1$, $y_i = -1$; The optimization problem of the primary of SVM is shown as follows:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, i = 1, 2, \ldots, m. \end{aligned} \tag{1}$$

C. Blockchain

Consortium Blockchain: The blockchain is essentially a public ledger running on a peer-to-peer network. This technique is famous for its capabilities of decentralization, transparency, reliability and so on [36]. Blockchain can be divided into three categories according to the degree of decentralization, the size of blockchain nodes and many other characteristics. They are public blockchain, consortium blockchain, and private blockchain.

1) The public blockchain is completely open. Users can join the blockchain network at any time to access the data recorded on the public ledger. The most typical application of public chains are Bitcoin and Ethereum.
2) The consortium blockchain is less open than the public blockchain. Only authenticated members can join the network and get access to the data recorded on the ledger. Fabric is the most popular consortium blockchain.
3) The private blockchain is generally applied within a single enterprise or organization. It is equipped with the lowest degree of openness, with a high level of access control and authority management.

From the classification of the blockchain, consortium blockchain agrees with the requirements of application across multiple organizations in terms of suitable degree of openness and high security.

Smart Contract: A smart contract is a set of code that runs on a blockchain. It can be called by the blockchain nodes, before automatic execution without human intervention. The emergence of smart contracts has enabled blockchain application extend from the digital currency field to other domains. With smart contracts, many traditional applications are able to remove the third party, greatly simplifying the running procedures of the centralized system, improving the operation efficiency and reducing the running cost.

D. Threshold Paillier

Threshold Paillier [27], [28] with additively homomorphic properties is an effective methods of constructing an SMC protocol. The homomorphic properties can be described as follows: $[[M_1 + M_2]] = [[M_1]][[M_2]]$ and $[[kM_1]] = [[M_1]]^k$ The following steps are the main operations of Threshold Paillier.

Key Generation: At the step of key generation, several pairs of keys are generated and each pair contains a public key and a sub-private key, where all the public keys are the same and the sub-private keys are the shares of the private key. Firstly, integer n is chosen which satisfies that $n = p * q$, where p and q are two strong primes. Based on these two primes, the private key sk is chosen. By the secure-sharing protocol [37], sk is shared among several participants, and the i-th of which achieves a sub-private key sk_i. Meanwhile, the public key pk composed of n, g and g^{sk} is sent to all participants as well, where $g = (1+n)^a b^m \bmod n^2$ and a, b are randomly selected values.

Encryption: Given a message m, the encryption function $c = g^m r^n \bmod n^2$ is used to encrypt m and get the encrypted value c. In this function, r is a randomly picked value.

Share Decryption: At this step, all the participants compute a decryption share by its own sk_i. For the i-th participant, function $c_i = c^{sk_i}$ is used for the calculation. Then, the decryption shares are sent to the curator.

Combination: If the number of decryption shares is less than the threshold t, the curator is unable to get the final decryption result. On the contrary, the decryption shares are combined to get the final decryption value.

IV. PROBLEM DESCRIPTION

In this section, we give a description of the system model. Based on the model, we build the threat model and establish design goals.

A. System Model

We divide our system into three components based on their relationships with the data. As shown in Fig. 1, they are VSNs

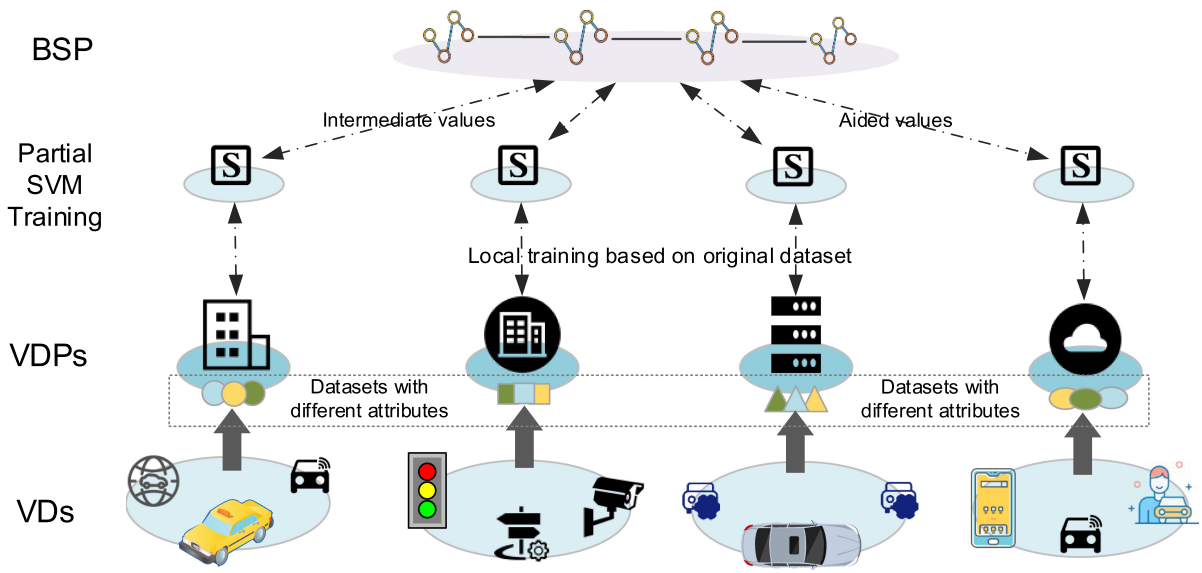


Fig. 1. System model.

device (VD), VSNs data provider (VDP) and blockchain service platform (BSP).

1) VSNs Device: Raw data in VSNs are from multiple VSNs devices such as vehicle sensors, mobile devices, roadside units. The raw data contains valuable information which helps VDPs to get an overview of the VSNs. The VSNs devices transmit a variety of data to VDPs every moment and the data is the basis of almost all VSNs applications.
2) VSNs Data Provider: VSNs data providers are the entities who collect, store and process raw VSNs data. Different data providers always have different types of data in volumes, sources, and especially attributes, this explains why they have to work together to train an efficient classifier. Thus we can also treat VSNs data providers as the SVM classifier trainer. At each iteration in the training process, most calculation operations are done locally using data with the attributes of their own. Then the intermediate values are recorded on the blockchain after being encrypted by Paillier.
3) Blockchain Service Platform: In our proposed scheme, a data sharing platform is established through blockchain to maintain a unified data interaction mechanism, which greatly improves efficiency. It is a service platform running on a consortium blockchain. On one hand, it provides VDPs with a decentralized and transparent data sharing platform which allows VDPs to acquire all the data recorded on BSP. Meanwhile, nobody can change the data recorded on BSP. On the other hand, BSP possesses strong security protection and keep the data invisible for entities outside VDPs. Moreover, the communication data between BSP and VDPs are also encrypted to prevent data disclosure.

B. Threat Model

In our scheme, when training an SVM classifier, there are several interactions between VDPs and BSP, and there exist potential threats among these interactions. In our security model, we consider VDPs are honest-but-curious. It means that all VDPs cooperate to train an SVM followed by the correct protocols, but some of them are curious about the information contained in others' shared data. There are two main types of threat models we pay attention to.

1) Known Ciphertext Model. The BSP is open for all VDPs, and thus each VDP has access to all the public data recorded in BSP. Those data include encrypted intermediate values of calculation, decrypted values using sub-secret keys of all VDPs.
2) Known Background Model. We assume that some of the VDPs collude to analyze the shared data on BSP. In this model, more valuable information may be obtained compared to the upper threat model.

C. Design Goals

Based on the system model and threat model, to satisfy the specific requirement of data sharing in VSNs, we aim to design a both efficient and privacy-preserving SVM classifier. The following are the design goals.

1) The data privacy can be well preserved. To achieve this goal, we have to prevent data disclosure from two aspects according to the threat model. Firstly when faced with the honest-but-curious adversaries, the data recorded on BSP has no risk of privacy disclosure. Secondly, if there exits VDPs colluding with each other, the privacy of data recorded on BSP is still confidential.
2) The classifier's high accuracy can be maintained. No negative impact should be brought because of the high security. To guarantee that the classifier is usable and useful, the accuracy of whether an SVM classifier trained by our scheme is as high as that trained by normal approaches should be our second design goal.
3) The low training overhead can be achieved. When training an SVM classifier in a secure way, the encryption operations and communication between participants are usually necessary. Considering the data volume in the real

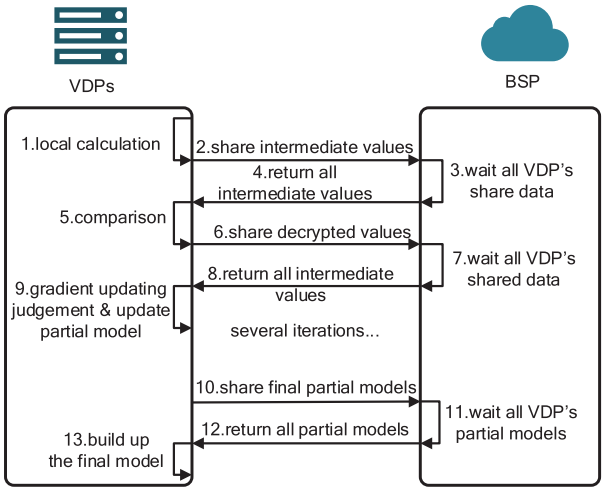


Fig. 2. System overview.

scene and the computational efficiency of homomorphic encryption itself, our proposed scheme should have low training overhead including less encryption calculations and low communication time overhead.

V. The Proposed Scheme

In this section, we present our proposed scheme in detail from five parts: system overview, system initialization, local training process, privacy-preserving gradient update judge algorithm and data sharing on BSP.

A. System Overview

To describe our proposed scheme in a clear way, we assume that three VDPs aim to train an SVM classifier together where the training dataset is combined by the data of themselves with different attributes. As shown in Fig. 2, at each iteration, each of the three VDPs trains a partial model locally using their own original data. Before computing the gradient, all VDPs have to encrypt their own intermediate values using the same public key and generate three random aided parameters.

Then, for each VDP, the encrypted intermediate values and the three processed random parameters are recorded in BSP to make all VDPs able to get access to them. Meanwhile, when the three VDPs acquire another two VDPs' shared data, they can judge how to update the three partial models securely and update the models. During one iteration, each VDP has only two encryption operations and one decryption operation.

Finally, after several iterations, all the VDPs record their own partial model parameters on BSP. By acquiring all the partial model parameters, each VDP is able to build up an integral model.

B. System Initialization

Our scheme is established on the threshold Paillier cryptosystem. Therefore before our scheme runs, three pairs of keys should be generated for three VDPS, where the public keys are the same and each sub-private key is a share of the private key. Existing threshold key management schemes can be introduced to help negotiate such multiple key pairs via secret-sharing protocols.

After that, to provide VDPs with a secure and reliable data sharing platform, three VDPs need to join the consortium blockchain system as a node so as to share data with the BSP. Any entity that wants to join the blockchain needs to be authenticated firstly, and unjoined entities cannot get access to the data recorded on the consortium blockchain ledger.

Finally, all the VDPs need to negotiate the parameters of their initial partial model before training the model, negotiate the label of each data instance and determine the order of all instances.

C. Local Training Process

In our proposed scheme, most training computations are carried out locally by the VDPs during the training process and the original data are kept locally. During one iteration, intermediate value sharing between VDPs is required to perform a comparison before the gradient is updated. The result of the comparison determines how the gradients are updated. After comparison, VDP calculates the gradients and updates their own partial model parameters. The following part explains the local training process.

1) Stochastic Gradient Descent for SVM: The SVM optimization algorithm based on stochastic gradient descent (SGD) is simple and efficient. Generally, only one training instance is used for per iteration and the running time is unrelated with the size of the training dataset, so the algorithm is especially suitable for big data analysis. In the VSNs, the size of data is large, and the data update frequently and stochastic gradient descent optimization methods can satisfy the application requirements. SVM based on stochastic gradient descent can be expressed in the following form:

$$f(w) = \frac{1}{2}w^T w + C\sum_{i=1}^{m}\max\left(0, 1 - y_i w^T x_i\right) \tag{2}$$

The right part of the equation is the hinge-loss function, where C is the misclassification penalty and $\frac{1}{m}$ is taken as its value.

At each iteration, we can calculate the gradient by formula as shown in Eq. (3).

$$\Delta_t = \lambda w_t - I\left[(wx_i < 1)\right] x_i y_i \tag{3}$$

If $I|(wx_i < 1)|$ is true, it means $(wx_i < 1), I[wx_i < 1)] = 1$; Otherwize, $I[(wx_i < 1)] = 0$.

Then we can update the w by Eq. (4).

$$w_{t+1} = w_t - \lambda\Delta_t \tag{4}$$

2) Partial SVM Classifier Training Based on SGD: In our scheme, the attributes of the training dataset consist of the attributes of all partial datasets provided by each VDP. During the SVM classifier training process based on SGD, we observe that most training calculations can be performed locally.

For example, an instance with three attributes x_i= [$attr_1$, $attr_2$, $attr_3$] is vertically-partitioned into three parts on average and we generate them into three VDPs A, B, C each

Algorithm 1: SVM Based on SGD.

Input: Training set D, learning rate λ, maxIters T.
Output: Trained model w^*.

for $t = 1$ to T **do**
  Select i_t from D randomly.
  Update Δ_{t+1} by Eq. (2).
  Update w_{t+1} by Eq. (3).
end for
return w^*.

Algorithm 2: Partial Model Training Process.

Input: Training set D^A, D^B, D^C, learning rate λ, maxIters T.
Output: Trained model w^*.

All VDPs perform the following operations simultaneously. Take VDP A to describe in detail.
for $t = 1$ to T **do**
  Select i_t randomly.
  Calculate $y \sum_{i=1}^{d^A} w_i x_i$.
  Cooperate with other VDPs to judge how to update gradient by Eq. (5).
  Update Δ_{t+1} by Eq. (2).
  Update w_{t+1} by Eq. (3).
end for
Get several partial model parameters and combine them.
return w^*.

of which holds $[attr_1]$, $[attr_2]$, $[attr_3]$. At the $t+1$ iteration, the model parameter is defined as $w_{t+1} = [v_1, v_2, v_3]$ where each of v_1, v_2, v_3 is seperately held by A, B and C. Before calculating the gradient, the training can be performed locally using A, B, C's own training data and model parameters. For VDP A, VDP B and VDP C, they calculate the intermediate values of $v_1 attr_1, v_2 attr_2$ and $v_3 attr_3$, respectively. Then we use Eq. (5) where $I[(wx_i < 1)]$ is defined to judge how to update the gradient. In the above case, the three intermediate values need to be shared between the three VDPs to calculate the sum of them and then compare the sum with 1. After the comparison, the gradient and the partial model parameters of each VDP are updated locally. We represent wx_i by a in the following sections.

Through one iteration of the training process, only when calculating I, data exchange between multiple VDPs is required. The rest of the training operations are performed locally.

$$I = \begin{cases} 1 & y_i \left(w^A x_i^A + w^B x_i^B + w^C x_i^C \right) < 1 \\ 0 & \text{otherwize} \end{cases} \tag{5}$$

After a predetermined number of iterations, each VDP obtains the final partial model, and they share partial models to build up an integral model which is the final SVM classifier. The complete content is described in Algorithm 2.

Obviously, this comparison operation requires the participation of three VDPs. In the process of privacy protection, we will also introduce the details in the following two sections.

Algorithm 3: Privacy-Preserving Gradient Update Judge.

Input A: $[[a^i]]$ from VDP i.
Input B: r_1^i, $[[r_2^i]]$, r_3^i from VDP i.
Output: $a > 1$ or $a < 1$.

Each VDP *i* picks three positive integers r_1^i, r_2^i, r_3^i, where $|r_3^i - r_2^i| < r_1^i$, and encrypts r_2^i to get $[[r_2^i]]$.
Each VDP *i* uploads $[[a^i]]$, r_1^i, $[[r_2^i]]$, r_3^i.
Each VDP *i* downloads all the other VDPs' $[[a^i]]$, r_1^i, $[[r_2^i]]$, r_3^i.
Each VDP *i* calculates $[[a]]$, $[[r_2]]$ by Eq. (6) and Eq. (7), and calculates r_1 and r_3 where $r_1 = \sum_{i=1}^n r_1^i$ and $r_3 = \sum_{i=1}^n r_3^i$.
Each VDP *i* calculates $[[ar_1 + r_2]]$ by Eq. (8).
Each VDP *i* decrypts $[[ar_1 + r_2]]$ by sub-private key SK^i and uploads it to BSP.
Each VDP *i* downloads all other decrypted values from VDPs to recover $(ar_1 + r_2)$, and compares $(ar_1 + r_2)$ with $(r_1 + r_3)$.
If $(ar_1 + r_2) > (r_1 + r_3)$, $a > 1$; Else $a < 1$.
return $a > 1$ or $a < 1$.

D. Privacy-Preserving Gradient Update Judge

To protect the privacy of intermediate values from being leaked, we implement a privacy-preserving gradient update judge scheme based on the threshold homomorphic encryption algorithm without introducing a third party. Each VDP hides the calculation results for each step through threshold Paillier. By threshold Paillier's homomorphic property, we can perform addition operations securely. To judge how to update the gradients, here we use additively homomorphic encryption to construct Eqs.(6), (7) and (8).

$$[[a]] = \left[\left[\sum_{i=1}^{n} a^i\right]\right] = \prod_{i=1}^{n} [[a^i]] \tag{6}$$

$$[[r_2]] = \left[\left[\sum_{i=1}^{n} r_2^i\right]\right] = \prod_{i=1}^{n} [[r_2^i]] \tag{7}$$

$$[[ar_1 + r_2]] = [[ar_1]][[r_2]] = \left[\left[\sum_{i=1}^{r_1} a\right]\right] [[r_2]]$$

$$= \prod_{i=1}^{r_1} [[a]][[r_2]] = [[a]]^{r_1}[[r_2]] \tag{8}$$

During judgment, the most critical step is to compare the encrypted calculation result with the constant 1. In Algorithm 3, we perform the comparison operation securely by three random numbers computed by random positive integers provided by each VDP.

The three partial random values selected by each VDP satisfy the condition described in Algorithm 3 and it is not hard to prove that the combined random value r_1, r_2, and r_3 still satisfy the condition. For integer a, if $(ar_1 + r_2) > (r_1 + r_3)$, we can derive that $a > 1$, otherwise $a < 1$.

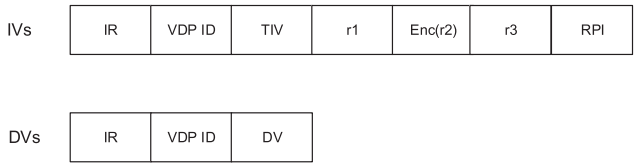


Fig. 3. Formats of IVs and DVs.

E. Data Sharing on BSP

The core function of BSP is to provide a secure and open data sharing platform for VDPs. BSP replaces the complex and direct communication between VDPs with the communication between VDPs and BSP by smart contracts. VDPs run blockchain nodes to share data and query data by calling smart contracts running on the blockchain. During an iterative process, each VDP needs to record data twice: its own intermediate values (IVs) and decrypted values (DVs). At the same time, the data recorded on the public ledger will be read twice, which are the IVs and DVs of other VDPs. The data formats of the two types of data recorded on the ledger are as shown in Fig. 3:

1) The Format of IVs

Iteration Round: During the training process, obviously there are several iterations at which the IVs are needed to share between VDPs. The IVs of each iteration are different. This field can mark the current training round of the model, avoiding using different rounds of data in one calculation. *Iteration Round* is maintained by smart contracts.

VDP ID: A unique identifier distinguishing a VDP from others. When a VDP node calls the smart contracts, its address is recorded in this field automatically.

Training Intermediate Value: It is encrypted by VDP before uploaded. The sum of all training intermediate values will be compared with 1 to determine how to update the model.

r_1*:* A random positive integer helps during the comparison process. It is unencrypted.

r_2*:* A random positive integer helps during the comparison process. It is encrypted.

r_3*:* A random positive integer helps during the comparison process. It is unencrypted.

Random Positive Integer: It is generated randomly between 1 and m which determines the data instances selected in the next iteration.

2) The Format of DVs

Iteration Round: Similar function described in IVs.

VDP ID: Similar function described in IVs.

Decrypted Value: The decrypted value represents the decrypted shares of all VDPs. The result of the final decryption result can be obtained by combining all of these values.

VI. Security Analysis

The most rigorous definition currently widely accepted is described in Goldreich's paper [38], which establishes a trusted third-party T as the ideal model that can communicate securely with other participants. Because the ideal model is in the most secure state, if the simulated output of the ideal model is indistinguishable from the real protocol, it proves that the real protocol has the same security as the ideal protocol. The adversary in the ideal model can conduct the same attack with the adversary in the real model.

In the ideal protocol, all the calculations are done by T, and there is no interaction between the n participants. All the information obtained by the attacker is the input and output of the corrupter. In the real protocol, since there is no trusted third party, the participants need to interact with each other. Therefore, in addition to obtaining the input and output of all the corrupters, the attacker also obtains interactive data of the corrupter in the calculation process. If we can prove that the data is worthless, it can show that the attacker gets the same information in the real and ideal protocols. By constructing a simulator S in the ideal model, all the intermediate data obtained by the attacker in the real protocol is simulated under the premise that all the corrupters' inputs and outputs are known. If the simulator can simulate the data in real protocols in polynomial time, it can prove that the attacker obtains the same information under the two models, and thus demonstrating security.

The computational security definition of a secure multi-party computing protocol is given below.

Definition 1: The multi-party computation protocol with n participants under the cryptography model is considered to be computational secure, if for any attacker A, there exists a corresponding simulator S in the ideal model interacting with A, and satisfying the following conditions:

1) The running time of S is the polynomial of A's running time.
2) For any input set, the $n+1$ outputs produced by the multi-party computation protocol are computationally indistinguishable from the $n+1$ outputs produced by the ideal model.

To prove the security of our proposed solution under the known ciphertext and background models, we conducted a security analysis based on the above idea. Thus we acquire the information which an attacker can get from the ideal model and the real protocol. Then we compare them and prove they are indistinguishable.

In our scheme, n VDPs are involved to share their encrypted intermediate values to calculate F: $F([[a]]^1, \ldots, [[a]]^n, 1, r_1^1, [[r_2^1]], r_3^1, \ldots, r_1^n, [[r_2^n]], r_3^n)$.

Assume that the attacker has corrupted a set of VDPs $A = P_{i1}, \ldots, P_{i|A|}$. Then all the data the attacker gets in the ideal model is the input $([[a]]^{i1}, \ldots, [[a]]^{i|A|}, 1, r_1^{i1}, [[r_2^{i1}]], r_3^{i1}, \ldots, r_1^{i|A|}, [[r_2^{i|A|}]], r_3^{i|A|})$ and the output F of the VDPs. We construct a simulator S that simulates all the data the attacker gets in the real model based on the data the attacker obtained in the ideal model. Firstly, we analyze the information that the attacker can get in the real protocol.

Input Phase: Since all the VDPs share their encrypted input $[[a]]^1, \ldots, [[a]]^n, 1, r_1^1, [[r_2^1]], r_3^1, \ldots, r_1^n, [[r_2^n]], r_3^n$, the attacker is able to get all of them. Especially for the corrupted VDPs, the attacker also gets $a^{i1}, \ldots, a^{i|A|}, r_2^1, \ldots, r_2^{|A|}$.

Computation Phase: At each step of the calculation phase, the attacker obtains data $[[x+y]]$ based on $[[x]]$ and $[[y]]$.

Output Phase: In the output phase, the attacker gets the result $F([[a]]^1, \ldots, [[a]]^n, 1, r_1^1, [[r_2^1]], r_3^1, \ldots, r_1^n, [[r_2^n]], r_3^n)$.

Then we construct the simulator S of the polynomial time. S takes $a^{i1}, \ldots, a^{i|A|}, r_2^1, \ldots, r_2^n$ and F as the input. The following step S0 simulates the information calclulated based on the input.

Step S0: S generates the encrypted data $[[a^{i1}]], \ldots, [[a^{i|A|}]]$, $[[r_2^{i1}]], \ldots, [[r_2^{i|A|}]]$ and $[[F]]$ based on $a^{i1}, \ldots, a^{i|A|}, r_2^1, \ldots, r_2^n$ and F. Then, S can simulate the calculations based on those encryted data such as $[[a^{i1} + a^{i2}]]$ and $[[r_2^{i1} + r_2^{i2}]]$.

Step S1: After step S0, we can simulate part of the calculated intermediate values which are defined as $[[a^{j1}]], \ldots, [[a^{j|r|}]], [[r_2^{j1}]], \ldots, [[r_2^{jr}]]$. Then for the remaining intermediate values that cannot be directly simulated by $S0$, S simulates them by selecting the random numbers to generate the corresponding ciphertext. According to the threshold cryptosystem's security, these simulations are successful.

Step S2: Based on steps S0 and S1, we can simulate all the values calculated in the computation phase.

Step S3: F is one of S's input, so S can easily get a simulation of F.

From the above simulation process, the information obtained by the attacker from the ideal model and the information obtained from the real model are computationally indistinguishable. The security of our scheme can be proven.

VII. Performance Evaluation

In this section, we conduct experiments to evaluate the performance of our scheme in terms of accuracy and efficiency. By the result of classification accuracy, we aim to verify that the privacy preserving methods introduced in our scheme have no negative effect on the classifier's accuracy. Meanwhile, how features related to efficiency affect the efficiency is analyzed, including the dataset, the number of participants and the number of iterations.

A. Experimental Settings

In our scheme, several participants cooperate to train a classifier with their own dataset securely. We implement the training operations for each participant by Java and Go. The Java program runs on a PC (AMD Ryzen 5 2600X Six-Core processor at 3.60 GHz and 32 GB RAM) to simulate the whole training process. The Go program runs as chaincode on the consortium blockchain (fabric 1.3) in a virtual machine set with 4 GB memory.

Threshold Paillier's operations are performed in the integer space, but there are a large number of floating point number calculations during the SVM training process. Therefore, to process the values in the original dataset, we have to represent the floating point values with an integer format. According to the international standard IEEE 754, any binary floating point number D can be expressed as $D = (-1)^S \times M \times 2^E$, where S represents a sign bit, M represents a significant number, and E represents an exponent bit. We use the same method to solve the floating point number problem in the program implementation. On the other hand, we set threshold Paillier's key N to 256 bits and the threshold t is set to be the number of all sub-private keys. To evaluate the performance of our scheme, we use the real-world datasets Breast Cancer Wisconsin Data (BCWD) [39] and Australian Credit Approval Data (ACAD) [40]. The detailed information about the two datasets is listed in Table II. To simulate the operation of the scheme under real-world scenarios, we partition every dataset vertically into three parts on average, each of which has one third of the attributes of the dataset on average and is held by a single VDP.

TABLE II
Statistics of Datasets

Datasets name	Instances number	Attributes number
BCWD	699	9
ACAD	690	14

TABLE III
Performance of Classifier Accuracy

Dataset	Precision		Recall	
	PP-SVM	SVM	PP-SVM	SVM
BCWD	91.60%	91.22%	99.58%	99.58%
ACAD	88.70%	91.76%	81.98%	81.46%

The parameters used to train the model are set as follows: the maximum number of iterations for training is 1500, the learning rate is 0.00095 and the number of samples selected from the training data set in each iteration is 1.

B. Accuracy

Two criterions (precision P and recall R) are adopted to evaluate the SVM classifier trained by our scheme. We can compute P by equation $P = t_p/(f_p + t_p)$ and compute R by equation $R = t_p/(f_n + t_p)$, where t_p is the amount of positive instances that are classified correctly, f_p is the amount of negative instances that are classified correctly and f_n is the amount of positive instances that are classified incorrectly.

We need to prove in the experiments that the accuracy of the SVM classifier under our scheme will not reduce. Thus in this experiment, we compare the result of the SVM classifier (PP-SVM) based on our privacy-preserving scheme with the SVM classifier (SVM) trained generally under the same training parameters.

As shown in Table III, we can observe our PP-SVM classifier sufferes no accuracy loss compared with general SVM classifiers. Theoretically, although threshold Paillier is introduced in our scheme, the encryption and decryption operations do not result in any calculation accuracy loss, which guarantees the final classifier's effectiveness.

C. Efficiency

We evaluate the efficiency of our proposed scheme from two criterions: time overhead and scalability. The evaluation results show whether our scheme is efficient and practical enough when applied into real-world scenarios.

TABLE IV
PERFORMANCE OF CLASSIFIER EFFICIENCY

Dataset	Total Time Overhead	Calculation Time Overhead		Communication Time Overhead	
	Total Time	Local Training Time	Gradient Update Judge Time	1-th Communication Time	2-th Communication Time
BCWD	384649ms	65ms	8064ms	186034ms	190486ms
ACAD	393261ms	58ms	8247ms	189332ms	195624ms

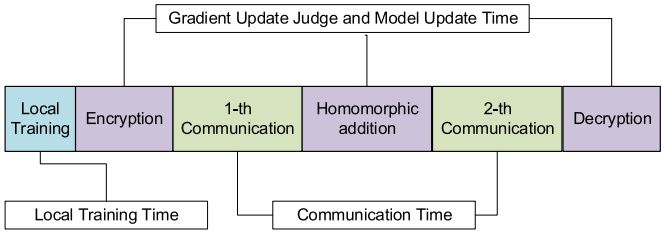


Fig. 4. Time calculation.

1) Time Overhead Evaluation: The total time overhead in our scheme mainly consists of two parts: calculation time overhead and communication time overhead, where the calculation time is related to local training time (LTT) and gradient update judge time (GUJT). The communication time is related to the latency of calling smart contracts to share and query data. To evaluate the total time consumption during the training process of our scheme, we carry out another experiment.

For the three VDPs involved in the experiment, they parallelly train the partial models over their own partial dataset during one iteration. During this process, there are no time-consuming operations such as encryption or decryption and we name the time cost in this process LTT. Then, essential encryption and decrytion operations are performed to update the gradient and the model. The calculation time cost during this process is named GUJT. Meanwhile, two communications are necessary to share the intermediate values. The detailed time cost information is shown in Fig. 4.

The results of time cost statistics are shown in Table IV. Over both datasets, the total time cost is low enough where the communication time overhead takes up a large proportion compared with calculation time overhead.

Although the training process can be finished in a short time, the communication time cost is still a little high. As we introduce consortium blockchain as the data sharing platform, extra time needs to be spent on consensus algorithms between nodes and it cannot be avoided.

2) Scalability Evaluation: In the above experiments, there are three VDPs training an SVM classifier. We conduct more experiments to evaluate the scalability of our scheme when the amount of VDPs increases. In these experiments, the dataset is divided into 3 to 5 parts vertically, and each VDP holds one of them. The result is shown in Figs. 5 and 6.

It can be concluded from the Fig. 5 that the number of VDPs has no effect on the classifier's accuracy. However, in Fig. 6, we observe that the increase of VDP amounts brings negative effects on calculation time overhead. Theoretically, compared to experiments with three VDPs, experiments with four VDPs spend more time on encrytion and decryption during the gradient

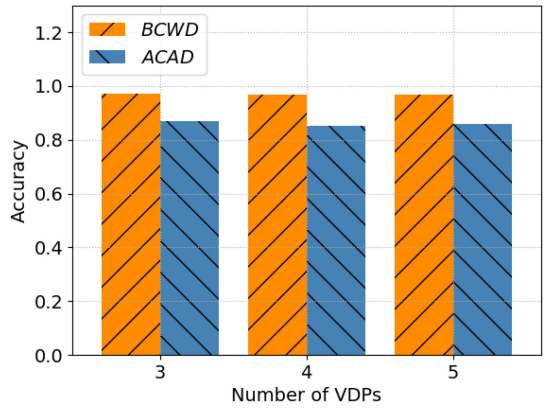


Fig. 5. Accuracy with different numbers of VDPs.

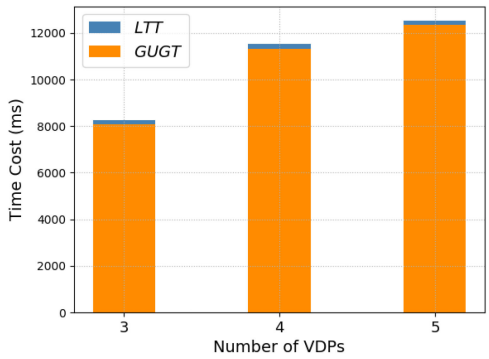


Fig. 6. Time consumption of dataset BCWD with different numbers of VDPs.

update judge process. Nevertheless, the total communication time does not have obvious increase. The way we define the communication time during a round of iterations is the time interval between the time when the data is first uploaded to the BSP and the time when the data is last queried by a VDP. On one hand, since the workload performed before each VDP uploads data is basically the same, all VDPs almost simultaneously upload data and receive the returned data. When the number of VDPs increases from 3 to 5, the communication time does not change much. For communication time, it mainly depends on the number of iterations which is fixed at 1500 in all the experiments.

VIII. CONCLUSION

In this paper, we proposed an efficient and secure SVM training scheme which helps multiple VSNs data providers to train an SVM classifier over vertically partitioned dataset together. Our scheme combines consortium blockchain techniques with threshold Paillier to build up a decentralized and secure SVM training platform. To achieve a high performance, most training operations are performed locally over the original data, only

a few intermediate values are essential to be shared on the platform. Extensive experiments are conducted and the results demonstrate that our scheme can train an accurate SVM classifier with low time cost. In the future work, we will focus on the optimization of communication overhead and the expansion of machine learning methods. Although the total time cost of the training process is not high in the current scheme, the communication time between VDPs and BSP is still worth optimizing. In addition, the current algorithm is only designed for SVM based on stochastic gradient descent. To cope with the expansion of machine learning algorithms in different scenarios, the future work will focus on creating a framework that supports multiple algorithms.

REFERENCES

[1] N. Cheng *et al.*, "Big data driven vehicular networks," *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, Nov./Dec. 2018.

[2] Y. Zhang *et al.*, "BDS: A centralized near-optimal overlay network for inter-datacenter data replication," in *Proc. 13th EuroSys Conf.*, ACM, 2018, p. 10.

[3] Z. Ning, F. Xia, N. Ullah, X. Kong, and X. Hu, "Vehicular social networks: Enabling smart mobility," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 16–55, May 2017.

[4] A. M. Vegni and V. Loscri, "A survey on vehicular social networks," *IEEE Commun. Surveys Tut.*, vol. 17, no. 4, pp. 2397–2419, Oct.–Dec. 2015.

[5] L. Lv *et al.*, "Communication-aware container placement and reassignment in large-scale internet data centers," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 540–555, Mar. 2019.

[6] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, "Efficient and privacy-preserving truth discovery in mobile crowd sensing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3854–3865, Apr. 2019.

[7] N. Kato *et al.*, "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 146–153, Jun. 2017.

[8] Z. M. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tut.*, vol. 19, no. 4, pp. 2432–2455, Oct.-Dec. 2017.

[9] G. Xu, H. Li, H. Ren, K. Yang, and R. H. Deng, "Data security issues in deep learning: Attacks, countermeasures and opportunities," *IEEE Commun. Mag.*, vol. 57, no. 11, pp. 116–122, Nov. 2019.

[10] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Trans. Ind. Inform.*, to be published, doi: 10.1109/ TII.2019.2945367.

[11] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, no. 1, pp. 911–926, 2020.

[12] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE S&P*, 2000, pp. 44–55.

[13] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, vol. 4324, p. 4325.

[14] Y. Li, Q. Luo, J. Liu, H. Guo, and N. Kato, "TSP security in intelligent and connected vehicles: Challenges and solutions," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 125–131, Jun. 2019.

[15] M. Shen, B. Ma, L. Zhu, X. Du, and K. Xu, "Secure phrase search for intelligent processing of encrypted data in cloud-based IoT," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1998–2008, Apr. 2019.

[16] H. Li, D. Liu, Y. Dai, T. H. Luan, and S. Yu, "Personalized search over encrypted data with efficient and secure updates in mobile clouds," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 97–109, Jan.–Mar. 2018.

[17] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 19–38.

[18] H. Li, Y. Yang, Y. Dai, J. Bai, S. Yu, and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 484–494, Apr.–Jun. 2020.

[19] P. Mohassel and P. Rindal, "ABY 3: A mixed protocol framework for machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, ACM, 2018, pp. 35–52.

[20] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order Markov chains and application attribute bigrams," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, pp. 1830–1843, Aug. 2017.

[21] K. Xu, H. Yue, L. Guo, Y. Guo, and Y. Fang, "Privacy-preserving machine learning algorithms for big data systems," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst.*, 2015, pp. 318–327.

[22] M. Shen, B. Ma, L. Zhu, R. Mijumbi, X. Du, and J. Hu, "Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 4, pp. 940–953, Apr. 2018.

[23] H. Ren, H. Li, Y. Dai, K. Yang, and X. Lin, "Querying in Internet of Things with privacy preserving: Challenges, solutions and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 144–151, Nov./Dec. 2018.

[24] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 334–348.

[25] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 4, pp. 870–885, Apr. 2019.

[26] 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 Things J.*, vol. 6, no. 5, pp. 7702-7712, Oct. 2019.

[27] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Boca Raton: Chapman and Hall/CRC, 2014.

[28] P.-A. Fouque, G. Poupard, and J. Stern, "Sharing decryption in the context of voting or lotteries," in *International Conference on Financial Cryptography*, Berlin, Germany: Springer, 2000, pp. 90–104.

[29] W. Jiang, H. Li, G. Xu, M. Wen, G. Dong, and X. Lin, "PTAS: Privacy-preserving thin-client authentication scheme in blockchain-based PKI," *Future Gener. Comput. Syst.*, vol. 96, pp. 185–195, 2019.

[30] M. Shen, Y. Deng, L. Zhu, X. Du, and N. Guizani, "Privacy-preserving image retrieval for medical IoT systems: A blockchain-based approach," *IEEE Netw.*, vol. 33, no. 5, pp. 27–33, Sep.–Oct. 2019.

[31] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, ACM, 2015, pp. 1310–1321.

[32] A. Gascón *et al.*, "Secure linear regression on vertically partitioned datasets," *IACR Cryptology ePrint Archive*, vol. 2016, p. 892, 2016.

[33] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, ACM, 2016, pp. 308–318.

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

[35] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[36] M. Iansiti and K. R. Lakhani, "The truth about blockchain," *Harvard Bus. Rev.*, vol. 95, no. 1, pp. 118–127, 2017.

[37] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[38] O. Goldreich, *Foundations of Cryptography: Basic Tools*. Cambridge, U.K.: Cambridge University, 2001.

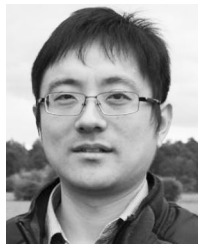
[39] O. L. Mangasarian and W. H. Wolberg, "Cancer diagnosis via linear programming," Department of Computer Sciences, University of Wisconsin-Madison, Tech. Rep., 1990.

[40] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

Meng Shen (M'14) received the B.Eng. degree from Shandong University, Jinan, China, in 2009, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2014, both in computer science. Currently he is with the Beijing Institute of Technology, Beijing, China, as an Associate Professor. His research interests include privacy protection for cloud and IoT, blockchain applications, and encrypted traffic classification. Dr. Shen received the Best Paper Runner-Up Award at IEEE IPCCC 2014.

Jie Zhang received the B.Eng. degree in computer science from the China University of Mining and Technology, Jiangsu, China, in 2018. Currently, he is working toward the master's degree with the Department of Computer Science, Beijing Institute of Technology, Beijing, China. His research interests include the application of blockchain and privacy-preserving machine learning.

Liehuang Zhu (M'16) is a Professor with the Department of Computer Science at the Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from the Ministry of Education, China. His research interests include Internet of Things, cloud computing security, Internet and mobile security.



Ke Xu received the Ph.D. degree from Tsinghua University, Beijing, China, where he serves as a Full Professor. He has published more than 100 technical articles and holds 20 patents in the research areas of next-generation Internet, P2P systems, Internet of Things (IoT), and network virtualization and optimization. He is a member of ACM and has guest edited several special issues in IEEE and Springer Journals. Currently, he is holding Visiting Professor position with the University of Essex, U.K.

Xiangyun Tang received the B.Eng. degree in computer science from the Minzu University of China, Beijing, China, in 2016. Currently she is working toward the Ph.D degree with the Department of Computer Science, Beijing Institute of Technology. Her research interests include differential privacy and security multi-party computation.