Secure Dot Product of Outsourced Encrypted Vectors and its Application to SVM

Jun Zhang
Department of Computer Science
The University of Hong Kong
Pokfulam Road, Hong Kong
jzhang3@cs.hku.hk

Xin Wang
Department of Computer Science
The University of Hong Kong
Pokfulam Road, Hong Kong
u3003679@connect.hku.hk

S. M. Yiu
Department of Computer Science
The University of Hong Kong
Pokfulam Road, Hong Kong
smyiu@cs.hku.hk

Zoe L. Jiang
Shenzhen Graduate School
Harbin Institute of Technology
Shenzhen, China
zoeljiang@gmail.com

Jin Li
School of Computer Science
Guangzhou University
Guangzhou, China
jinli71@gmail.com

ABSTRACT

It is getting popular for users to outsource their data to a cloud system as well as leverage the high-speed computing power of this third-party platform to process the data. For the sake of data privacy, outsourced data from different users is usually encrypted under different keys. To enable users to run data mining algorithms collaboratively in the cloud, we need an efficient scheme to process the encrypted data under multiple keys. Dot product is one of the most important building blocks of data mining algorithms. In this paper, we show how to give the cloud the permission to decrypt the encrypted dot product of two encrypted vectors without compromising the privacy of the data owners. We propose the first feasible scheme that trains a SVM (Support Vector Machine) classifier for both horizontally and vertically partitioned datasets using only one server. Existing schemes either can only handle a much simpler classifier (linear mean classifier) with two non-colluding servers or can only be applied to vertically partitioned dataset. We also show that our scheme not only preserves data privacy but also runs faster than existing schemes.

Keywords

Secure Dot Product, Outsourced Partitioned Data, Integer Vector Encryption, SVM

1. INTRODUCTION

As cloud computing is gaining popularity, an increasing number of users tend to outsource their data to a cloud system and leverage the high-speed computing power to process their data [20]. However, the privacy of the outsourced data is a major concern [23]. It is common for users to encrypt their data before outsourcing it to the cloud. This motivates the study of privacy-preserving data mining algorithms, i.e., how the cloud system executes data mining procedures on encrypted data. On the other hand, in order to maximize the efficiency of data mining, companies or researchers would like to collaborate and combine their databases without disclosing information of each other's database. Data might be horizontally (different records with same attributes), vertically (same records with different attributes) partitioned among the users. For example, each branch of a company owns part of a customer database (horizontally partitioned). Due to the privacy agreement, they are not allowed to disclose any information of individual customers to other branches except using the information for statistical analysis. The encrypted records are all stored in the same cloud system. The cloud system would help the company to data-mine the combined database for marketing purposes. Besides that the data is in encrypted form, an additional difficulty of this problem is records are encrypted under different keys. Dot product is a basic operation in many data mining algorithms, such as SVM, k-means clustering and etc. In this paper, we focus on designing a secure dot product protocol and propose one scheme to train a SVM classifier (one of the most popular classifiers) for both horizontally and vertically partitioned datasets.

SCC'17, April 02 2017, Abu Dhabi, United Arab Emirates
© 2017 ACM. ISBN 978-1-4503-4970-3/17/04...$15.00
DOI: http://dx.doi.org/10.1145/3055259.3055270

1.1 Existing work

Within the context of privacy preserving data mining, several secure dot product protocols [7, 8, 25, 26, 30, 13, 3, 11, 2, 9, 17] have been proposed. Secure dot protocol acts as an important building block in association rule mining [25], Naïve Bayes classifier [26] and decision tree classifier [8] on vertically partitioned data, Bayesian network [30] on distributed data, K-means clustering [13] on arbitrarily partitioned data, as well as neural network [3] and SVM [14, 27]. The schemes proposed by Du et al. [7] and Vaidya et al. [25] have been proved to be insecure in a state-of-the-art overview [11]. Besides, a privacy preserving dot product protocol on horizontally partitioned data was proposed in [2]. However, all the aforementioned papers are under a distributed model (see Fig. 1a). Each user not only stores the

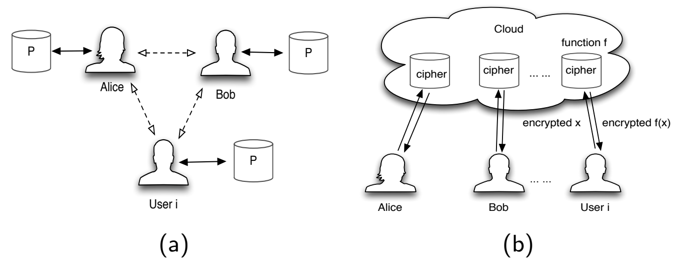

Figure 1: Two collaborative data mining models. (a) The distributed model; (b) The outsourced model.

data locally but also computes locally, and interacts with others in order to finish the data mining task. In the setting of data outsourced to the cloud as shown in Fig. 1b, Liu et al. [9, 17] proposed to use BGN cryptosystem [6] which supports homomorphic additions and one homomorphic multiplication under single key to compute the dot product. However, the scheme in [9] requires all the users to stay online and the one in [17] is in essence a single key protocol.

In theory, fully homomorphic encryption (FHE) [10] and Yao's garbled circuits can compute the dot product when there is only one user (single key). There also exists a FHE scheme for multiple keys [18]. However, it is still not efficient enough for practical use. To enable computation on outsourced data under multiple keys, several research work has been done [21, 28, 29, 16]. The key idea of [21] is to use two non-colluding servers to transform all the ciphertexts under different keys into those under the same key, thus incurring a huge amount of interactions between the servers. Wang et al. [28] reduced the communication overhead using proxy re-encryption [4]. Furthermore, Wang et al. [29] improved their own approach using a different encryption scheme to make it secure even if a user colludes with a server. In both [28] and [29], partially homomorphic encryption schemes are used for efficiency purpose. But then, if the underlying encryption is additively homomorphic, the two servers need to jointly compute multiplication and vice versa. In all the above schemes, if the two servers collude, they can reveal all the data. Thus, they all rely on *two non-colluding* servers. Also, these schemes only work on linear mean classifier, a much simpler classifier than SVM. And it can be shown that the schemes are not secure when applied to SVM (see the attack in the Appendix).

For SVM classifier, only [16] considered it. They assume that the database is vertically partitioned among the users. They proposed to use FHE to solve the problem with only *one* server to avoid the problem of collusion between the servers. There are a few issues in this scheme. (i) FHE is slow. (ii) It requires the users to be online. (iii) One user needs to download and decrypt the whole gram matrix (see the next section for the definition) to compute the classifier. For n training samples, the size of gram matrix is n^2, which is a heavy communication overhead between the user and the cloud. And (iv), the scheme cannot be applied to horizontally partitioned data as it is not secure.

1.2 Our approach and contributions

To summarize, we need to resolve two issues in order to get a better scheme: (i) to avoid using FHE and partially homomorphic encryption and (ii) to achieve the effect of proxy re-encryption without using two servers. Recently, Zhou et al. [32] developed a cryptosystem for integer vectors that supports addition, linear transformation and weighted dot product. However, it is restricted to single user. Fortunately, an important technique in [32] is "key-switching" that allows a user to change a ciphertext to another ciphertext under a different key without decrypting the original ciphertext. To do this, technically we need to generate a "key-switching matrix". In our setting for multiple users (keys), giving this matrix to the cloud would reveal the secret keys of users. We solve this problem based on the notion of "key-switching" sub-matrix inspired by the the notion of block matrix in linear algebra. Using the integer vector encryption and the above idea, we show how to compute dot product on horizontally or vertically partitioned data from multiple users using different encryption keys in one single cloud system. Moreover, we construct a well-designed protocol for SVM classifier training and classification. Our scheme has the following properties. (i) We only require one server. (ii) It works on horizontally, vertically partitioned and hybrid database. (iii) Users do not need to be online.

2. PRELIMINARIES

2.1 The System Model

We focus on the outsourced model shown as Fig. 1b. The involved parties are multiple users and the cloud.

(1) User (U): there are multiple users in our model. Different users will encrypt their data using different keys, then upload the encrypted records to the cloud.

(2) Cloud (C): the cloud acts as a storage and computing platform. It will try to run data mining algorithms on the ciphertexts. For example, if the involved parties are interested in SVM, the cloud will train SVM on the ciphertexts and obtain an encrypted classifier for future classification.

2.2 The Threat Model

We assume that both the cloud and the users are honest but curious. All will execute the protocol correctly, but try to infer sensitive information. And there is no collusion between the users and the cloud. We assume the cloud observes only the encrypted records without knowing anything about the unencrypted records.

2.3 Notations

We use small bold letters for vectors and capital bold letters for matrices (i.e, $\mathbf{x_i}$ is a column vector). The integer vector encryption is denoted as $\mathcal{E}(\ )$. The maximum absolute value of the entries in a vector $\mathbf{v}$ or a matrix $\mathbf{M}$ is denoted as $|\mathbf{v}|$ or $|\mathbf{M}|$. For a vector $\mathbf{x}$, $\lceil \mathbf{x} \rfloor$ represents rounding each element of $\mathbf{x}$ to the nearest integer. For a matrix $\mathbf{M}$, $vec(\mathbf{M})$ denotes a column vector by concatenating all the columns of $\mathbf{M}$.

2.4 Support Vector Machine

We are given a training data set $\{(\mathbf{x_i}, y_i)\}_{i=1}^{n}$, where the training sample $\mathbf{x_i} \in R^m$ (column vector[1]) and the class label $y_i \in \{+1, -1\}$. We consider a *linear* binary classification task, i.e., SVM finds a best separating hyperplane

[1]Note that in the database point of view, each training record is a row, but in SVM, it is represented as a column.

$(\mathbf{w^T x} - b = 0)$ to partition data into two classes according to the label, where $\mathbf{w} \in R^m$ is a weight vector and b is a bias item. The margin is $\frac{1}{||\mathbf{w}||}$. In order to cope with noise, we use *"soft" margin* in practice, of which ξ denotes the slack variable to allow error. To maximize the margin while minimizing the error, the standard SVM solution is formulated as follows [19].

$$\begin{aligned} &\min_{\mathbf{w},\xi} \frac{1}{2}\mathbf{w^T w} + C\sum_{i=1}^{n} \xi_i \\ &s.\ t. \quad y_i(\mathbf{w^T x_i} - b) \geq 1 - \xi_i \ and\ \xi_i \geq 0\ \forall i. \end{aligned} \tag{1}$$

where C is used to balance the margin size and the error. The primal form of SVM is often solved in its dual form.

$$\begin{aligned} &\min_{\alpha} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha^T y_i y_j \mathbf{K}(\mathbf{x_i}, \mathbf{x_j})\alpha - \sum_{i=1}^{n}\alpha_i \\ &s.\ t. \quad 0 \leq \alpha_i \leq C\ and\ \sum_{i=1}^{n}\alpha_i y_i = 0 \end{aligned} \tag{2}$$

where $\alpha \in R^n$ and $\mathbf{K}(\mathbf{x_i}, \mathbf{x_j})$ is called a kernel. The *kernel matrix* $\mathbf{K} \in R^{n\times n}$ contains the kernel values for every pair of training samples. For linear SVM, $\mathbf{K}$ is called *gram matrix* computed as $\mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i^T x_j}$. After we get α by solving (2), we can further compute $\mathbf{w} = \sum_{i=1}^{n}\alpha_i \mathbf{x_i} y_i$. To compute the bias item b, we pick sample $\mathbf{x_i}$ whose corresponding coefficient $\alpha_i > 0$ and compute $b = \mathbf{w^T x_i} - y_i$. For the sake of robustness, we can also use all the support vectors to compute b and take the average. Finally, the separating plane is $\mathbf{w^T x} - b = 0$.

2.5 Integer Vector Encryption

We describe the integer vector encryption [31] in this section. Instead of encrypting every element in the vector, it encrypts the vector as a whole and performs key-switching, which transforms the ciphertext $\mathbf{c_1}$ under key $\mathbf{S_1}$ to the ciphertext $\mathbf{c_2}$ under any chosen secret key $\mathbf{S_2}$.

Integer Vector Encryption

Encryption: Given an integer vector $\mathbf{x} \in Z_p^m$, where m is the vector length, p is the alphabet size and $\mathbf{S} \in Z^{m\times n}$ is the secret key, the ciphertext $\mathbf{c} = \mathcal{E}(\mathbf{x}) \in Z^n$ is a vector that satisfies $\mathbf{Sc} = t\mathbf{x} + \mathbf{e}$, where t is a large integer and e is an error term with elements smaller than $\frac{t}{2}$.
Decryption: With secret key S, decryption can be done as $\mathbf{x} = \lceil \frac{\mathbf{Sc}}{\mathbf{t}} \rfloor$.
Key-switching: The goal is to transform the ciphertext $\mathbf{c}$ under the key $\mathbf{S}$ to a ciphertext $\mathbf{c}' \in Z^{n'}$ under a different key $\mathbf{S}' \in Z^{m\times n'}$ such that $\mathbf{S}'\mathbf{c}' = \mathbf{Sc}$. Firstly $\mathbf{c}$ and $\mathbf{S}$ are converted into an intermediate representation $\mathbf{c}^* \in Z^{n\times l}$ and $\mathbf{S}^* \in Z^{m\times(n\times l)}$ (l is an integer such that $|\mathbf{c}| < 2^l$). $\mathbf{c}^*$ is the binary representation of $\mathbf{c}$. By keeping $|\mathbf{c}^*| = 1$, we can control the increase in error. Besides, we should make sure that $\mathbf{S}^*\mathbf{c}^* = \mathbf{Sc}$. Therefore, the construction of $\mathbf{S}^*$ should include a multiplier for each bit in $\mathbf{c}^*$. In the second step, we constructs a *"key-switching matrix"* $\mathbf{M} \in Z^{n'\times(n\times l)}$ which satisfies (3), and the new ciphertext is $\mathbf{c}' = \mathbf{Mc}^*$.

$$\begin{aligned} &\mathbf{S}'\mathbf{M} = \mathbf{S}^* + \mathbf{E} \\ s.\ t.\quad &\mathbf{S}'\mathbf{c}' = \mathbf{S}'\mathbf{Mc}^* = \mathbf{S}^*\mathbf{c}^* + \mathbf{Ec}^* \end{aligned} \tag{3}$$

where $\mathbf{E} \in Z^{m\times(n\times l)}$ is a random noise matrix, $\mathbf{e}' = \mathbf{Ec}^*$ is the new error and must be kept small. Since $|\mathbf{c}^*| = 1$, $\mathbf{e}'$ is kept small as long as $\mathbf{E}$ is randomly generated with $|\mathbf{E}|$ small. $\mathbf{S}'$ is not a square matrix, there is no simple way to compute $\mathbf{M} = (\mathbf{S}')^{-1}(\mathbf{S}^* + \mathbf{E})$ on the integer domain. Therefore, the secret key $\mathbf{S}'$ is restricted to a simple form $\mathbf{S}' = [\mathbf{I}, \mathbf{T}]$, where $\mathbf{I} \in Z^{m\times m}$ is an identity matrix concatenated with matrix $\mathbf{T} \in Z^{m\times(n'-m)}$. Given that $\mathbf{A} \in Z^{(n'-m)\times(n\times l)}$ is a random matrix, $\mathbf{M}$ can be computed as

$$\mathbf{M} = \begin{bmatrix} \mathbf{S}^* - \mathbf{TA} + \mathbf{E} \\ \mathbf{A} \end{bmatrix} \tag{4}$$

Addition: Given two plaintext-ciphertext pairs under the same key - $(\mathbf{x_1}, \mathbf{c_1})$ and $(\mathbf{x_2}, \mathbf{c_2})$, we compute $\mathbf{c_1} + \mathbf{c_2}$ to get $\mathcal{E}(\mathbf{x_1} + \mathbf{x_2})$. If $\mathbf{c_1}$ and $\mathbf{c_2}$ are under different keys, we should perform key-switching first to change them to those under the same key and then compute the addition.

$$\mathbf{S}(\mathbf{c_1} + \mathbf{c_2}) = t(\mathbf{x_1} + \mathbf{x_2}) + (\mathbf{e_1} + \mathbf{e_2}) \tag{5}$$

Dot Product: Given two plaintext-ciphertext pairs under different keys - $(\mathbf{x_1}, \mathbf{c_1})$ under key $\mathbf{S_1}$, $(\mathbf{x_2}, \mathbf{c_2})$ under key $\mathbf{S_2}$. We have

$$\begin{cases} \mathbf{S_1 c_1} = t\mathbf{x_1} + \mathbf{e_1} \\ \mathbf{S_2 c_2} = t\mathbf{x_2} + \mathbf{e_2} \end{cases} \tag{6}$$

The encrypted dot product of $\mathbf{x_1}^T\mathbf{x_2}$ is computed as $\mathcal{E}(\mathbf{x_1}^T\mathbf{x_2}) = \lceil \frac{vec(\mathbf{c_1 c_2}^T)}{t} \rfloor$ whose secret key is $vec(\mathbf{S_1}^T\mathbf{S_2})^T$.

---

3. OUR SCHEME

The above integer vector encryption is only suitable for single-user scenario. Considering the implementation of dot product, although $\mathbf{S_1}$ and $\mathbf{S_2}$ are different keys, they must belong to the same user and only this particular user can derive the decryption key to get the dot product in plaintext. In this section, we demonstrate how to allow the cloud to decrypt the encrypted dot product when multiple users are involved. For ease of understanding, assume that we have two users - Alice owns vector $\mathbf{x_a}$ under key $\mathbf{S_a}$ and Bob has vector $\mathbf{x_b}$ under key $\mathbf{S_b}$. We consider two cases. One is that the cloud wants to know the dot product of vectors from one user (i.e., $\mathbf{x_a}^T\mathbf{x_a}$). The other is that the cloud tries to get the dot product of vectors from different users (i.e., $\mathbf{x_a}^T\mathbf{x_b}$).

3.1 Dot Product of Vectors from the Same User

If two vectors are from the same user (Alice), a simple case is $\mathbf{x_a}^T\mathbf{x_a}$. According to the integer vector encryption, the cloud needs $vec(\mathbf{S_a}^T\mathbf{S_a})^T$ to decrypt $\mathcal{E}(\mathbf{x_a}^T\mathbf{x_a})$. A straightforward way is to reveal $\mathbf{S_a}^T\mathbf{S_a}$ to the cloud. If $\mathbf{S_a}$ is completely random, then it is secure to release $\mathbf{S_a}^T\mathbf{S_a}$. For example, given that we have an orthogonal matrix $\mathbf{A}$ with $\mathbf{A}^T\mathbf{A} = \mathbf{I}$, even if we know the product is $\mathbf{I}$, it is still infeasible to get $\mathbf{A}$. But in our case, all the operations are executed on the integer domain. Each secret key is restricted to the form of $[\mathbf{I}, \mathbf{T}]$, where $\mathbf{I}$ is an identity matrix concatenated with some matrix $\mathbf{T}$. Thus, $\mathbf{S_a}^T\mathbf{S_a}$ can be computed as

$$\mathbf{S_a}^T\mathbf{S_a} = \begin{bmatrix} \mathbf{I} \\ \mathbf{T_a}^T \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{T_a} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{T_a} \\ \mathbf{T_a}^T & \mathbf{T_a}^T\mathbf{T_a} \end{bmatrix} \tag{7}$$

An attacker can deduce the original secret key $\mathbf{T_a}$ of Alice from $\mathbf{S_a}^T\mathbf{S_a}$. To solve this problem, we use the key-switching technique to transform $\mathcal{E}(\mathbf{x_a}^T\mathbf{x_a})$ from key $vec(\mathbf{S_a}^T\mathbf{S_a})^T$ to secret key $[\mathbf{I}, \mathbf{T_c}]$ of the cloud. We can compute $\mathbf{M_a}$ as (8), where $\mathbf{A_a}$ is a random matrix, and $\mathbf{E_a}$ is a random noise matrix.

$$\mathbf{M_a} = \begin{bmatrix} (vec(\mathbf{S_a}^T\mathbf{S_a})^T)^* - \mathbf{T_c}\mathbf{A_a} + \mathbf{E_a} \\ \mathbf{A_a} \end{bmatrix} \tag{8}$$

3.2 Dot Product of Vectors from Different Users

Recall that Alice owns vector $\mathbf{x_a}$ under key $\mathbf{S_a}$ and Bob has vector $\mathbf{x_b}$ under key $\mathbf{S_b}$. $\mathbf{x_a}$ and $\mathbf{x_b}$ are two vectors from different users. The cloud needs $\mathbf{S_a}^T\mathbf{S_b}$ to decrypt $\mathcal{E}(\mathbf{x_a}^T\mathbf{x_b})$. Similarly, the reveal of $\mathbf{S_a}^T\mathbf{S_b}$ poses risk to the secret keys of Alice and Bob. Again, we depend on the key-switching technique to change the key of $\mathcal{E}(\mathbf{x_a}^T\mathbf{x_b})$ from $vec(\mathbf{S_a}^T\mathbf{S_b})^T$ to $[\mathbf{I}, \mathbf{T_c}]$, which is the secret key of the cloud. The computation of key-switching matrix $\mathbf{M_{ab}}$ requires the secret keys of both Alice and Bob. However, Alice and Bob knows nothing about each other's secret key. Therefore, neither Alice nor Bob can work out the key-switching matrix alone. Instead, we let the cloud get the final key-switching matrix through secret aggregation. Inspired by the notion of block matrix in linear algebra, we divide $\mathbf{S_a}^T\mathbf{S_b}$ into three parts and we compute key-switching matrix $\mathbf{M_{ab}}$ through breaking it into the addition of three key-switching sub-matrices.

$$\begin{aligned} \mathbf{S_a}^T\mathbf{S_b} &= \begin{bmatrix} \mathbf{I} & \mathbf{T_b} \\ \mathbf{T_a}^\mathbf{T} & \mathbf{T_a}^T\mathbf{T_b} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{T_a}^T & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{T_b} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T_a}^T\mathbf{T_b} \end{bmatrix} \\ &= \mathbf{S_{ab_1}} + \mathbf{S_{ab_2}} + \mathbf{S_{ab_3}} \end{aligned} \tag{9}$$

Alice owns $\mathbf{T_a}$, Bob owns $\mathbf{T_b}$ and the cloud can get $\mathbf{T_a}^T\mathbf{T_b}$ through randomization. Firstly, Alice generates a random matrix $\mathbf{P_a} \in Z^{(n'-m)\times(n'-m)}$ and sends $\mathbf{P_a}\mathbf{T_a}^T$ to Bob. Next, Bob generates a random matrix $\mathbf{P_b} \in Z^{(n'-m)\times(n'-m)}$ and sends $\mathbf{P_a}\mathbf{T_a}^T\mathbf{T_b}\mathbf{P_b}$ to the cloud. Alice and Bob sends $\mathbf{P_a}$, $\mathbf{P_b}$ to the cloud respectively. Then the cloud computes the inverse of $\mathbf{P_a}$ and $\mathbf{P_b}$ to recover $\mathbf{T_a}^T\mathbf{T_b}$ through computing $\mathbf{P_a}^{-1}\mathbf{P_a}\mathbf{T_a}^T\mathbf{T_b}\mathbf{P_b}\mathbf{P_b}^{-1}$. Therefore Alice, Bob and the cloud hold $\mathbf{S_{ab_1}}$, $\mathbf{S_{ab_2}}$ and $\mathbf{S_{ab_3}}$ separately and they can compute key-switching sub-matrices $\mathbf{M_{ab_1}}$, $\mathbf{M_{ab_2}}$ and $\mathbf{M_{ab_3}}$ separately as follows. We choose three random matrices $\mathbf{A_{ab_i}}$ and random error matrices $\mathbf{E_{ab_i}}$ for $i \in \{1, 2, 3\}$. Assume that the maximum of error matrix $\mathbf{E}$ in (4) is $|\mathbf{E}|$, then the maximum of $\mathbf{E_{ab_i}}$ should satisfy $|\mathbf{E_{ab_i}}| \leq \frac{|\mathbf{E}|}{3}$.

$$\mathbf{M_{ab_i}} = \begin{bmatrix} (vec(\mathbf{S_{ab_i}})^T)^* - \mathbf{T_c}\mathbf{A_{ab_i}} + \mathbf{E_{ab_i}} \\ \mathbf{A_{ab_i}} \end{bmatrix} \tag{10}$$

Finally, we get the key-switching matrix $\mathbf{M_{ab}} = \sum_{i=1}^{3} \mathbf{M_{ab_i}}$. As we choose different random matrix $\mathbf{A_{ab_i}}$ and different random error matrix $\mathbf{E_{ab_i}}$, we can make sure that the release of $\mathbf{M_{ab_i}}$ reveals nothing about $\mathbf{T_a}$ or $\mathbf{T_b}$.

3.3 SVM Training and Classification on Horizontally Partitioned Data

Based on the aforementioned techniques of computing dot product of two vectors, we show how to train a SVM classifier on horizontally partitioned data. Supposed that the cloud stores n_a encrypted samples uploaded by Alice, n_b encrypted samples uploaded by Bob, where $n = n_a + n_b$. The encrypted gram matrix $\mathcal{E}(\mathbf{K})$ with $\mathcal{E}(\mathbf{K_{i,j}}) = \mathcal{E}(\mathbf{x_i}^T\mathbf{x_j})$ can be calculated as follows.

$$\left[\begin{array}{ccc|ccc} \mathcal{E}(\mathbf{x_1}^T\mathbf{x_1}) & \cdots & \mathcal{E}(\mathbf{x_1}^T\mathbf{x_{n_a}}) & \mathcal{E}(\mathbf{x_1}^T\mathbf{x_{n_a+1}}) & \cdots & \mathcal{E}(\mathbf{x_1}^T\mathbf{x_n}) \\ \mathcal{E}(\mathbf{x_2}^T\mathbf{x_1}) & \cdots & \mathcal{E}(\mathbf{x_2}^T\mathbf{x_{n_a}}) & \mathcal{E}(\mathbf{x_2}^T\mathbf{x_{n_a+1}}) & \cdots & \mathcal{E}(\mathbf{x_2}^T\mathbf{x_n}) \\ \vdots & \text{I} & \vdots & \vdots & \text{II} & \vdots \\ \mathcal{E}(\mathbf{x_{n_a}}^T\mathbf{x_1}) & \cdots & \mathcal{E}(\mathbf{x_{n_a}}^T\mathbf{x_{n_a}}) & \mathcal{E}(\mathbf{x_{n_a}}^T\mathbf{x_{n_a+1}}) & \cdots & \mathcal{E}(\mathbf{x_{n_a}}^T\mathbf{x_n}) \\ \hline \mathcal{E}(\mathbf{x_{n_a+1}}^T\mathbf{x_1}) & \cdots & \mathcal{E}(\mathbf{x_{n_a+1}}^T\mathbf{x_{n_a}}) & \mathcal{E}(\mathbf{x_{n_a+1}}^T\mathbf{x_{n_a+1}}) & \cdots & \mathcal{E}(\mathbf{x_{n_a+1}}^T\mathbf{x_n}) \\ \vdots & \text{III} & \vdots & \vdots & \text{IV} & \vdots \\ \mathcal{E}(\mathbf{x_n}^T\mathbf{x_1}) & \cdots & \mathcal{E}(\mathbf{x_n}^T\mathbf{x_{n_a}}) & \mathcal{E}(\mathbf{x_n}^T\mathbf{x_{n_a+1}}) & \cdots & \mathcal{E}(\mathbf{x_n}^T\mathbf{x_n}) \end{array}\right]$$

If dividing $\mathcal{E}(\mathbf{K})$ into four areas - $\text{I}, \text{II}, \text{III}, \text{IV}$, we observe that area I contains the dot products between encrypted samples of Alice. Areas II and III are dot products between encrypted samples of Alice and Bob. Area IV involves Bob's encrypted samples. Notice that $\mathcal{E}(\mathbf{K})$ is symmetric, thus we only need to compute and decrypt those encrypted dot products above the diagonal of $\mathcal{E}(\mathbf{K})$. According to the integer vector encryption, the required secret keys to decrypt $\mathcal{E}(\mathbf{K})$ are $vec(\mathbf{S_a}^T\mathbf{S_a})^T$, $vec(\mathbf{S_b}^T\mathbf{S_b})^T$ and $vec(\mathbf{S_a}^T\mathbf{S_b})$. With reference to section 3.1 and 3.2, the cloud can use $\mathbf{M_a}$, $\mathbf{M_b}$ and $\mathbf{M_{ab}}$ to transform the encrypted dot products to those under its own secret key. Therefore, the cloud can decrypt $\mathcal{E}(\mathbf{K})$ and get the gram matrix $\mathbf{K}$ for linear SVM. As for non-linear SVM using kernels such as RBF kernel $\mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) = exp(-\frac{||\mathbf{x_i}-\mathbf{x_j}||^2}{g})$ and polynomial kernel $\mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i}\mathbf{x_j} + 1)^p$, their kernel matrices can be computed from the gram matrix. For example, the RBF kernel can be decomposed to dot products in the gram matrix $\mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) = exp(-\frac{||\mathbf{x_i}-\mathbf{x_j}||^2}{g}) = exp(-\frac{||\mathbf{x_i}\mathbf{x_i}-2\mathbf{x_i}\mathbf{x_j}+\mathbf{x_j}\mathbf{x_j}||^2}{g})$. Therefore, our proposed scheme can also be used to solve non-linear SVM. We do not go into details here.

We show how to build a linear SVM classifier in **Protocol 1** after the cloud knows $\mathbf{K}$. Given that the class labels are stored in plaintext, the cloud can solve the dual optimization problem with $\mathbf{K}$ according to (2). Once α is known, the cloud will calculate the encrypted weight vector $\mathcal{E}(\mathbf{w})$ of the supporting plane as $\mathcal{E}(\mathbf{w}) = \sum_{i=1}^{n} \alpha_i \mathcal{E}(\mathbf{x_i}) y_i$ (see Section 2.4). With Alice's and Bob's encrypted samples, the cloud can compute $\mathcal{E}(\mathbf{w_a}) = \sum_{i=1}^{n_a} \alpha_i \mathcal{E}(\mathbf{x_i}) y_i$ and $\mathcal{E}(\mathbf{w_b}) = \sum_{i=n_a+1}^{n} \alpha_i \mathcal{E}(\mathbf{x_i}) y_i$. As $\mathcal{E}(\mathbf{w_a})$ and $\mathcal{E}(\mathbf{w_b})$ are encrypted under different keys, the cloud cannot add them together directly. Again, we resort to the key-switching technique. Firstly, Alice and Bob agrees on a secret output key $\mathbf{S_{output}}$, which is hidden from the cloud. Next, Alice and Bob generate their key-switching matrices and upload to the cloud. Then the cloud uses the key-switching matrices to transform the key of $\mathcal{E}(\mathbf{w_a})$ and $\mathcal{E}(\mathbf{w_b})$ to $\mathbf{S_{output}}$. Finally, the cloud gets the encrypted weight vector through $\mathcal{E}(\mathbf{w}) = \mathcal{E}(\mathbf{w_a}) + \mathcal{E}(\mathbf{w_b})$.

In order to compute the bias item b, the cloud selects an encrypted training sample $\mathcal{E}(\mathbf{x_i})$ whose $\alpha_i > 0$. Supposed that it belongs to Alice, the cloud will compute $\mathcal{E}(\mathbf{w}^T\mathbf{x_i})$ and send it together with the class label y_i to Alice. Since Alice knows both the secret key of $\mathcal{E}(\mathbf{w})$ and $\mathcal{E}(\mathbf{x_i})$, she can decrypt $\mathcal{E}(\mathbf{w}^T\mathbf{x_i})$ to get $\mathbf{w}^T\mathbf{x_i}$ and calculate $b = \mathbf{w}^T\mathbf{x_i} - y_i$. Alice will send b to Bob based on one symmetric encryption. In our scheme, all the users will store b locally rather than upload to the cloud, because b is a not a vector. For the sake of robustness, the cloud can choose to use all the support vectors to compute b and take the average.

Protocol 1: SVM Training

Input: For the cloud: Encrypted training samples $\{\mathcal{E}(\mathbf{x_i})\}_{i=1}^{n}$ outsourced by Alice and Bob, class labels $\{y_i\}_{i=1}^{n}$, secret key S_c, key-switching matrices $\mathbf{M_a}$, $\mathbf{M_b}$ and $\mathbf{M_{ab}}$.

Output: The cloud gets the encrypted weight vector $\mathcal{E}(\mathbf{w})$ of the supporting plane ($\mathbf{w}^T\mathbf{x} - b = 0$). Each user stores b locally.

1. Encrypted Gram Matrix $\mathcal{E}(\mathbf{K})$: As gram matrix $\mathbf{K}$ is symmetric, we only compute the encrypted dot products above the diagonal. The cloud computes $\mathcal{E}(\mathbf{K})$ as $\mathcal{E}(\mathbf{K}_{ij}) = \mathcal{E}(\mathbf{x_i}^T\mathbf{x_j})$.
2. Ciphertext Transformation: Transform $\mathcal{E}(\mathbf{K}_{ij})$ to what the cloud can decrypt using its own secret key S_c. To be specific, the cloud will calculate $\mathbf{M_a}\mathcal{E}(\mathbf{K}_{ij})^*$ if $\mathbf{K}_{ij}$ is the dot product of two samples from Alice[2]. In a similar way, if the two samples are from Bob, the cloud will compute $\mathbf{M_b}\mathcal{E}(\mathbf{K_{ij}})^*$. If one sample is from Alice and the other is from Bob, the cloud will compute $\mathbf{M_{ab}}\mathcal{E}(\mathbf{K_{ij}})^*$.
3. Gram Matrix K: the cloud decrypts the transformed $\mathcal{E}(\mathbf{K})$ using S_c to get $\mathbf{K}$ in plaintext.
4. Compute α: Incorporate $\mathbf{K}$ into (2) and solve the dual optimization problem to obtain $\{\alpha\}_{i=1}^n$.
5. Encrypted weight vector $\mathcal{E}(\mathbf{w})$: the cloud computes $\mathcal{E}(\mathbf{w_a}) = \sum_{i=1}^{n_a}\alpha_i\mathcal{E}(\mathbf{x_i})y_i$ and $\mathcal{E}(\mathbf{w_b}) = \sum_{i=n_a+1}^{n}\alpha_i\mathcal{E}(\mathbf{x_i})y_i$. Alice and Bob agree on a secret output key $\mathbf{S_{output}}$ and then generate key-switching matrices separately to transform the key of $\mathcal{E}(\mathbf{w_a})$ and $\mathcal{E}(\mathbf{w_b})$ to $\mathbf{S_{output}}$. The cloud gets $\mathcal{E}(\mathbf{w}) = \mathcal{E}(\mathbf{w_a}) + \mathcal{E}(\mathbf{w_b})$.
6. Bias item b: Find one $\alpha_i > 0$ and select the corresponding encrypted sample $\mathcal{E}(\mathbf{x_i})$. Provided that this sample comes from Alice, the cloud will send $\mathcal{E}(\mathbf{w}^T\mathbf{x_i})$ and y_i to Alice. Alice uses $vec(\mathbf{S_a}^T\mathbf{S_{output}})^T$ to decrypt $\mathcal{E}(\mathbf{w}^T\mathbf{x_i})$ and compute $b = \mathbf{w}^T\mathbf{x_i} - y_i$. In the end, Alice sends b to Bob based on symmetric encryption.

When it comes to classification using the SVM classifier in the cloud, it is straightforward that we need to compute $\mathbf{w}^T\mathbf{x} - b$ and compare it with zero. In step 1 of Protocol 2, the cloud computes $\mathcal{E}(\mathbf{w}^T\mathbf{x})$ and sends it to the user that launches this query. In step 2, the user will get $\mathbf{w}^T\mathbf{x}$ and compares it with b. If $\mathbf{w}^T\mathbf{x} > b$ then the class label is $+1$, otherwise it is -1.

Protocol 2: SVM Classification

Input: For the cloud: An encrypted test sample $\mathcal{E}(\mathbf{x_j})$, encrypted weight vector $\mathcal{E}(\mathbf{w})$.
Output: Class label.
1. the cloud computes $\mathcal{E}(\mathbf{w}^T\mathbf{x_j})$ and returns it to the data owner.
2. The data owner will decrypt the received ciphertext $\mathcal{E}(\mathbf{w}^T\mathbf{x_j})$ and compare with b to get the class label.

3.4 Extensions

Horizontally partitioned dataset among multiple users: In section 3.3, we demonstrated how to train a svm classifier on horizontally partitioned dataset of two users. When extending to multiple users, the users must interact peer to peer at the set up phase to produce key-switching matrices.

Vertically partitioned dataset between multiple users: Compared to horizontal partitioning, it is much easier to cope with vertically partitioned data. The cloud can compute the gram matrix for each vertical partition and add them together at the end.

[2]Recall that $*$ denotes the binary representation.

Hybrid dataset between multiple users: The data set is first partitioned horizontally or vertically, then each partition can be recursively partitioned. The computation can be done recursively as well.

4. SECURITY ANALYSIS

In theorem 1, we prove the security of key-switching. we demonstrate that the cloud cannot deduce the old secret key from the key-switching matrix or the new key.

DEFINITION 1. *Learning with Errors (LWE) Problem: Given many samples of* $(\mathbf{a_i}, b_i) \in Z_q^m \times Z_q$*, it is infeasible to recover* $\mathbf{v} \in Z_q^m$ *from* $b_i = \mathbf{v}^T\mathbf{a_i} + \epsilon_i$ *with non-negligible probability.*

THEOREM 1. *It is infeasible to recover* $\mathbf{S}^*$ *from (3)* $\mathbf{S}'\mathbf{M} = \mathbf{S}^* + \mathbf{E}$ *given access to* $\mathbf{M}$ *and* S'*.*

Proof: We treat each element in Z to be elements of Z_q for some prime $q \gg max\{|\mathbf{S}'|, |\mathbf{M}|, |\mathbf{S}^*|, |\mathbf{E}|\}$. Since $\mathbf{S}'$ and $\mathbf{M}$ are known, we use $\mathbf{b_i}'$ to denote the i_{th} row of $\mathbf{S}'\mathbf{M}$. Similarly, we use $\mathbf{s_i}^*$ and $\mathbf{e_i}$ to represent the i_{th} row of $\mathbf{S}^*$ and $\mathbf{E}$ respectively. Thus we have $\mathbf{b_i}' = \mathbf{s_i}^* + \mathbf{e_i}$. Provided that $\mathbf{a_j}'$ is an identity vector, the equation $b'_{ij} = s_i^*a'_j + e_{ij}$ still holds. It shows that (3) can be reduced to the LWE problem. Hence, if we are able to compute S^* from $\mathbf{S}'\mathbf{M} = \mathbf{S}^* + \mathbf{E}$ given access to $\mathbf{M}$ and $\mathbf{S}'$, we can also construct a solver for the LWE problem, which contradicts with the fact that LWE problem is hard.

THEOREM 2. *With reference to theorem 1, the generation of key-switching matrix can be proved to be secure.*

Proof: According to theorem 1, the generation of key-switching matrices $\mathbf{M_a}$ and $\mathbf{M_b}$ is secure. When it comes to the generation of $\mathbf{M_{ab}}$, as $\mathbf{P_a}$ and $\mathbf{P_b}$ are random matrices, the cloud learns nothing about $\mathbf{T_a}$ or $\mathbf{T_b}$ from $\mathbf{P_a}^{-1}\mathbf{P_a}\mathbf{T_a}^T\mathbf{T_b}\mathbf{P_b}\mathbf{P_b}^{-1}$. Moreover, every key-switching sub-matrix can be generated securely as long as matrix $\mathbf{A_{ab_i}}$ and error matrix $\mathbf{E_{ab_i}}$ are random. Therefore, the generation of $\mathbf{M_{ab}}$ is also secure.

Then we analyze the security of Protocol 1 by evaluating the security of each step separately according to the Composition Theorem defined in [12]. If every step in Protocol 1 is secure, we can prove that the entire protocol is secure. The security of Protocol 1 under the assumption that the cloud is curious-but-honest can be defined as

THEOREM 3. *The cloud C learns nothing more than the result of gram matrix. It is computationally infeasible for C to infer any information about the users' original data* $\mathbf{x_i}$ *as long as the Integer Vector Encryption is semantically secure.*

Proof: We discuss the security of each step in Protocol 1.

Step 1 We analyze it with the Real and Ideal paradigm [12]. We need to prove that dot product of any two cipertexts is secure against a honest-but-curious ($\mathcal{HBC}$) adversary $\mathcal{A}_C^{\mathcal{HBC}}$ corrupting C in the real world. Besides, we build a simulator $\mathcal{F}^{\mathcal{HBC}}$ in the ideal world to simulate the view of the honest-but-curious adversary $\mathcal{A}_C^{\mathcal{HBC}}$. The view of $\mathcal{A}_C^{\mathcal{HBC}}$ includes its input $\{\mathcal{E}(\mathbf{x_i}), \mathcal{E}(\mathbf{x_j})\}$ and the output $\mathcal{E}(\mathbf{x_i}^T\mathbf{x_j})$. Simulator $\mathcal{F}^{\mathcal{HBC}}$ computes $\mathcal{E}(\mathbf{x_I}), \mathcal{E}(2\mathbf{x_I})$, where $\mathbf{x_I}$ is an identity vector and $2\mathbf{x_I}$ is also a vector of which each element is 2. Then $\mathcal{F}^{\mathcal{HBC}}$ computes $\mathcal{E}(\mathbf{x_I}^T2\mathbf{x_I})$ and returns $\{\mathcal{E}(\mathbf{x_I}), \mathcal{E}(2\mathbf{x_I}), \mathcal{E}(\mathbf{x_I}^T2\mathbf{x_I})\}$ to $\mathcal{A}_C^{\mathcal{HBC}}$. Since the view of $\mathcal{A}_C^{\mathcal{HBC}}$ are ciphertexts generated under Integer Vector Encryption and $\mathcal{A}_C^{\mathcal{HBC}}$ has no knowledge of the private key S. If

$\mathcal{A}_{\mathcal{C}}^{\mathcal{HBC}}$ can distinguish the view from the real world and the one from the ideal world, then it indicates $\mathcal{A}_{\mathcal{C}}^{\mathcal{HBC}}$ could succeed to distinguish ciphertexts generated by Integer Vector Encryption, which contradicts to the assumption that Integer Vector Encryption is semantically secure. Therefore, $\mathcal{A}_{\mathcal{C}}^{\mathcal{HBC}}$ is computationally infeasible to distinguish from the real world and the ideal world.

Step 2 We can prove the security of this step using the same method as Step 1. The view of $\mathcal{A}_{\mathcal{C}}^{\mathcal{HBC}}$ is $\{\mathbf{M_a}\mathcal{E}(\mathbf{K_{ij}})^*\}$, $\{\mathbf{M_b}\mathcal{E}(\mathbf{K_{ij}})\}$ and $\{\mathbf{M_{ab}}\mathcal{E}(\mathbf{K_{ij}})^*\}$. The simulator $\mathcal{F}^{\mathcal{HBC}}$ computes $\{\mathbf{M_a}\mathcal{E}(\mathbf{x_I}^T 2\mathbf{x_I})^*\}$, $\{\mathbf{M_b}\mathcal{E}(\mathbf{x_I}^T 2\mathbf{x_I})^*\}$,$\{\mathbf{M_{ab}}\mathcal{E}(\mathbf{x_I}^T 2\mathbf{x_I})^*\}$. The ciphertexts in the view of $\mathcal{A}_{\mathcal{C}}^{\mathcal{HBC}}$ cannot be distinguished.

Step 3 The plaintext value of gram matrix $\mathbf{K}$ is revealed in this step. We prove its security in Theorem 4.

Step 4 The cloud obtains α through solving a quadratic optimization problem. There are no sensitive training samples involved in this process. Hence this step is secure.

Step 5 The computation of $\mathcal{E}(\mathbf{w})$ runs on the encrypted training samples. Based on the semantic security of the integer vector encryption, the derivation of $\mathcal{E}(\mathbf{w_a})$ and $\mathcal{E}(\mathbf{w_b})$ is secure. According to theorem 1, it is secure to transform the key of $\mathcal{E}(\mathbf{w_a})$ and $\mathcal{E}(\mathbf{w_b})$ to $\mathbf{S_{output}}$. Accordingly, it is still secure to compute $\mathcal{E}(\mathbf{w}) = \mathcal{E}(\mathbf{w_a}) + \mathcal{E}(\mathbf{w_b})$.

Step 6 The bias item is computed as $b = \mathcal{E}(\mathbf{w}^T\mathbf{x_i}) - y_i = 0$. In our scheme, $\mathcal{E}(\mathbf{w}^T\mathbf{x_i})$ is decrypted locally by the data owner of $\mathbf{x_i}$. Consequently, this step is also secure.

THEOREM 4. *Under our threat model (see section 2.2), the attacker cannot recover any data record* $\mathbf{x_i}$.

Proof: Provided that the attacker knows the dot product $\mathbf{K_{ij}}$ of two vectors $\mathbf{x_i} \in Z^m$ and $\mathbf{x_j} \in Z^m$, there are $2m$ unknown variables and m linear equations, which is infeasible.

THEOREM 5. *The cloud cannot deduce the class label of the testing sample, thus Protocol 2 is secure.*

Proof: Recall that the cloud stores the encrypted weight vector $\mathcal{E}(\mathbf{w})$ for the SVM classifier. Given a testing sample $\mathcal{E}(\mathbf{x_j})$, the cloud will compute $\mathcal{E}(\mathbf{w}^T\mathbf{x_j})$ and send it to the data owner. Consequently, only the user owner of $\mathcal{E}(\mathbf{x_j})$ can get the final class label, while the cloud has no idea about the classification result.

5. EXPERIMENTAL EVALUATION

5.1 Performance of Our Scheme

We use Math Kernel Library (MKL) [1] to accelerate our implementation. The configuration of our PC is Ubuntu 14.04 64 bit operating system with Intel Core(TM) i7 CPU(2 cores), 2.2G Hz, and 8GB memory. Assume that Alice owns 40 training samples and Bob has 60 training samples, we show the running time of encryption, key-switching matrix generation and dot product computation in Fig. 2 with sample dimension varying from 10 to 50. As shown in Fig. 2, Alice and Bob can finish encryption and key-switching matrix generation efficiently. Then we analyze the time distribution of computing gram matrix on Alice's ciphertexts, Bob's training ciphertexts and both respectively[3]. Dot product operation will run $\frac{(1+40)\times 40}{2} = 820$ times on Alice's samples. Similarly, dot product operation is executed 1830 times on Bob's samples. Considering dot product of one sample from Alice and the other from Bob, 2400 operations will be executed. The time consumption of dot product (Alice) is the minimum, dot product (Bob) comes the second and finally dot product (both). We also measure the total time, which refers to the time consumed to compute the gram matrix.

[3]Time distribution on area I, II and IV above the diagonal (see section 3.3).

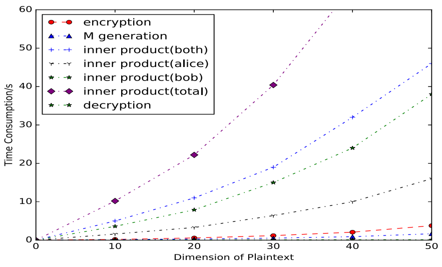

Figure 2: Performance of Our Scheme

5.2 Comparison with Existing Schemes

Only [9] and [17] are designed to compute dot product using the same one server model as ours, but there is no performance evaluation in [9] and [17] is in fact a prototype of [16]. Therefore, we compare our scheme with [16] and [29]. Firstly, we encrypt 3.6×10^5 samples of dimension 10 on our PC. The encryption time is 69 s, but [16] needs 360 s. Their experimental platform (8x Intel Xeon CPU, 3.6 GHz) is also better than us. As a result, integer vector encryption performs better than FHE, which is consistent with the test results (see section 5.1.2) of [31]. In [16], Alice needs to do additional n^2 encryptions (where n is the number of training samples), $2n^2$ decryptions to obtain the gram matrix. Bob needs to do n^2 encryptions and n^2 decryptions. Besides, the cloud needs to do additional $2n^2$ encryptions. However, Alice, Bob and the cloud need not to compute these additional encryptions or decryptions in our scheme. For 1000 samples of dimension 10, it takes about 3 hours to get the gram matrix in [16]. By contrast, it only takes 16 min in our scheme. The communication overhead of our scheme is mainly caused by the upload of key-switching matrices. No matter how many training samples we have, the communication overhead is constant and relies on the dimension m of the training sample and dimension T of the secret key $[\mathbf{I}, \mathbf{T}]$. In Fig. 3a, we fix $T = 10$ and vary the dimension m of training samples from 10 to 500. The size of key-switching matrix $\mathbf{M}$ changes from several MB to 1.4 GB. In Fig. 3b, we fix the dimension of m at 100 and vary the dimension of T from 10 to 50. We observe that the size of $\mathbf{M}$ varies from 60 MB to 550 MB. In [16], the communication overhead of Alice or Bob is $O(n^2)$. If T is fixed, our scheme incurs $O(m^2)$ communication overhead. We often have $m \ll n$ in practice. Therefore, our scheme performs better when taking into account the communication overhead.

To compare with the two non-colluding server model [29], we also run our scheme to train a linear mean classifier and record the time consumed. Given 50 samples of dimension 10, we need 1 s to transform the ciphertexts. Training a linear mean classifier requires $O(nm)$ additions and $O(m)$ multiplications. Using the integer vector encryption, linear mean classifier can be trained in 0.4 s. It takes 3.17 s to transform the ciphertexts and 0.63 s to train the classifier in

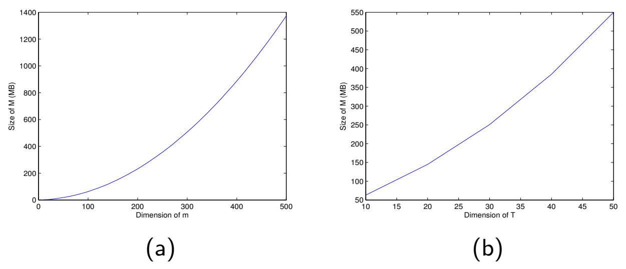

Figure 3: Communication overhead between a user and the cloud. (a) Overhead when m varies; (b) Overhead when T varies.

[29]. Therefore, our scheme achieves a similar performance as [29]. In terms of communication overhead, there is almost no communication cost during set up phase in [29]. However, it incurs heavy server-to-server interactions especially when addition and multiplication are executed many times.

6. DISCUSSION

To lower the memory requirement, we can use our scheme to solve a reduced SVM [15], which computes a reduced kernel matrix of size $n \times \bar{n}$ ($\bar{n} \ll n$) but has comparable classification accuracy to traditional SVM. Regardless of the inspiring improvements in SVM research such as Sequential Minimum Optimization (SMO) [22], core vector machine (CVM) [24], it is hard to implement on encrypted data not mentioning if outsourced to the cloud and multiple users are involved. For example, SMO picks out two training samples at each iteration, encryption makes it difficult to choose those samples violating the KKT conditions. On the other hand, CVM is proposed to reduce the training time complexity and space complexity. We cannot decide whether an encrypted training point falls outside of the ball During the classification phase, histogram intersection kernel SVM achieves run time complexity that is logarithmic in the number of support vectors instead of linear for the traditional SVM. Nevertheless, it requires sorting the support vectors which cannot be achieved on encrypted data under different keys. Moreover, according to a recent cryptanalysis [5] of the integer vector encryption, it is vulnerable to three attacks (broadcast encryption attack, chosen ciphertext attack and related chosen plaintext attack). The first attack does not apply to our scheme. When the attackers can only observe the encrypted records without knowing anything about the unencrypted records, the attacker cannot launch the other two attacks as well. Consequently, our scheme is secure under the threat model defined in section 2.2. We admit that it is a trade-off between efficiency and security.

7. CONCLUSION

In this paper, we studied the integer vector encryption and extended its application scenario to multiple users. We focus on the dot product operation supported by this encryption scheme, which is one of the most important building blocks among data mining algorithms. To facilitate the cloud to run data mining algorithms on encrypted data outsourced by multiple users, we allow the cloud to know the dot product of two encrypted vectors. We showed how to transform the encrypted dot product to what the cloud can decrypt based on key-switching matrix technique. It is worth mentioning that we put forward the notion of key-switching submatrix. Moreover, we proposed a privacy preserving scheme for linear SVM on partitioned (horizontally, vertically, or hybrid) data among multiple users. We also proved the security of our protocol and showed that our protocol is more efficient than the FHE scheme as well as comparable in performance to the two non-colluding server scheme. One limitation of our scheme is that it only works for integer (or fixed-point number with scaling) domain although it covers a wide range of applications already. Designing a scheme which supports operations on real domain is one of our future work. Moreover, we will explore solutions to other privacy preserving data mining algorithms based on the secure dot product proposed in this paper.

8. ACKNOWLEDGEMENT

The first three authors are partially supported by Seed Fund for Basic Research of HKU (201411159142, 201511159034) and the Collaborative Research Fund (CRF) of RGC, Hong Kong (C1008-16G). Zoe Jiang is supported by National Natural Science Foundation of China (No. 61402136), Guangdong Provincial Key Laboratory of High Performance Computing (No. [2013]82).

9. REFERENCES

[1] Intel math kernel library. https://software.intel.com/en-us/intel-mkl. Accessed Feb 1, 2016.

[2] A. Amirbekyan and V. Estivill-Castro. A new efficient privacy-preserving scalar product protocol. In *Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70*, pages 209–214. Australian Computer Society, Inc., 2007.

[3] 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*, pages 146–151. ACM, 2006.

[4] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in CryptologyÃćÂĂÂŤEUROCRYPT'98*, pages 127–144. Springer, 1998.

[5] S. Bogos, J. Gaspoz, and S. Vaudenay. Cryptanalysis of a homomorphic encryption scheme. In *ArcticCrypt 2016*, number EPFL-CONF-220692, 2016.

[6] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography Conference*, pages 325–341. Springer, 2005.

[7] W. Du and M. J. Atallah. Privacy-preserving cooperative statistical analysis. In *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, pages 102–110. IEEE, 2001.

[8] W. Du and Z. Zhan. Building decision tree classifier on private data. In *Proceedings of the IEEE international conference on Privacy, security and data mining-Volume 14*, pages 1–8. Australian Computer Society, Inc., 2002.

[9] L. Fang, W. K. Ng, and W. Zhang. Encrypted scalar product protocol for outsourced data mining. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 336–343. IEEE, 2014.

[10] C. Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

[11] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *Information Security and Cryptology–ICISC 2004*, pages 104–120. Springer, 2004.

[12] O. Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2004.

[13] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 593–599. ACM, 2005.

[14] S. Laur, H. Lipmaa, and T. Mielikäinen. Cryptographically private support vector machines. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 618–624. ACM, 2006.

[15] Y.-J. Lee and O. L. Mangasarian. Rsvm: Reduced support vector machines. In *SDM*, volume 1, pages 325–361. SIAM, 2001.

[16] F. Liu, W. K. Ng, and W. Zhang. Encrypted svm for outsourced data mining. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 1085–1092. IEEE, 2015.

[17] F. Liu, W. K. Ng, and W. Zhang. Secure scalar product for big-data in mapreduce. In *Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on*, pages 120–129. IEEE, 2015.

[18] A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234. ACM, 2012.

[19] O. L. Mangasarian and E. W. Wild. Proximal support vector machine classifiers. In *Proceedings KDD-2001: Knowledge Discovery and Data Mining*. Citeseer, 2001.

[20] P. Mell and T. Grance. The nist definition of cloud computing. 2011.

[21] A. Peter, E. Tews, and S. Katzenbeisser. Efficiently outsourcing multiparty computation under multiple keys. *Information Forensics and Security, IEEE Transactions on*, 8(12):2046–2058, 2013.

[22] J. Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[23] D. Song, E. Shi, I. Fischer, and U. Shankar. Cloud data protection for the masses. *Computer*, (1):39–45, 2012.

[24] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. In *Journal of Machine Learning Research*, pages 363–392, 2005.

[25] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644. ACM, 2002.

[26] J. Vaidya and C. Clifton. Privacy preserving naïve bayes classifier for vertically partitioned data. In *SDM*, pages 522–526. SIAM, 2004.

[27] J. Vaidya, H. Yu, and X. Jiang. Privacy-preserving svm classification. *Knowledge and Information Systems*, 14(2):161–178, 2008.

[28] B. Wang, M. Li, S. S. Chow, and H. Li. Computing encrypted cloud data efficiently under multiple keys. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 504–513. IEEE, 2013.

[29] B. Wang, M. Li, S. S. Chow, and H. Li. A tale of two clouds: Computing on data encrypted under multiple keys. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 337–345. IEEE, 2014.

[30] R. Wright and Z. Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–718. ACM, 2004.

[31] A. Yu, W. L. Lai, and J. Payor. Efficient integer vector homomorphic encryption. https://courses.csail.mit.edu/6.857/2015/files/yu-lai-payor.pdf, 2015.

[32] H. Zhou and G. Wornell. Efficient homomorphic encryption on integer vectors and its applications. In *Information Theory and Applications Workshop (ITA), 2014*, pages 1–9. IEEE, 2014.

APPENDIX

A. ATTACKS ON PREVIOUS SCHEMES

We refer the attack as *linear equation attack* which means that a user or the cloud tries to infer another user's sample via solving a series of linear equations.

(a) In [16], one of the users (e.g. Alice) has to download and decrypt the whole encrypted gram matrix. Given the gram matrix (dot products) and Alice's own samples, she tries to infer Bob's samples. If each user holds the same set of records but with different attributes (i.e. the database is vertically partitioned among the users), Alice can get the dot product of any two samples of Bob without knowing their exact value. However, when [16] is used to handle with horizontally partitioned data (e.g. users own different sets of records, but with all the attributes), Alice can construct m (the dimension of one sample) linear equations using m samples of Alice (as the coefficients) and the corresponding dot products to calculate the sample of Bob. Therefore, [16] cannot be applied to horizontally partitioned data.

(b) Both [28] and [29] work on linear mean classifier, a much simpler classifier than SVM. Both of them are not secure anymore if working on SVM on horizontally partitioned data. Since each user gets one copy of the gram matrix in plaintext at the end of the protocol, the linear equation attack (e.g. described in (a)) still exists.

In order to solve the optimization problem of SVM, the gram matrix must be revealed to one user or the cloud. Therefore, once user-server collusion occurs, the linear equation attack is inevitable on horizontally partitioned data. It is also the reason why we assume there is no collusion between the cloud and any user in model description.