Privacy-Preserving Elastic Net for Data Encrypted by Different Keys - With an Application on Biomarker Discovery

Jun Zhang, Meiqi He, and Siu-Ming Yiu(✉)

Department of Computer Science, The University of Hong Kong,
Pokfulam Road, Pok Fu Lam, Hong Kong
{jzhang3,mqhe,smyiu}@cs.hku.hk

Abstract. Elastic net is a popular linear regression tool and has many important applications, in particular, finding genomic biomarkers for cancers from gene expression profiles for personalized medicine (elastic net is currently the most accurate prediction method for this problem). There is an increasing trend for organizations to store their data (e.g. gene expression profiles) in an untrusted third-party cloud system in order to leverage both its storage capacity and computational power. Due to the privacy concern, data must be stored in its encrypted form. While there are quite a number of privacy-preserving data mining protocols on encrypted data, there does not exist one for elastic net. In this paper, we propose the first privacy-preserving elastic net protocol using two non-colluding servers. Our protocol is able to handle expression profiles encrypted from multiple medical units using different encryption keys. Thus, collaboration between multiple medical units are made possible without jeopardizing the privacy of data records. We formally prove that our protocol is secure and implemented the protocol. The experimental results show that our protocol runs reasonably fast, thus can be applied in practice.

Keywords: Privacy-preserving elastic net · Multiple encryption keys · Encrypted gene expression profiles · Biomarker discovery

1 Introduction

We motivate our study based on the following biomarker discovery application. The current cancer treatment based on doctors' empirical knowledge can be described as "one-size-fits-all" - almost all the patients diagnosed with the same cancer will receive similar treatment. Under this situation, some patients are likely to be under-treated while others be over-treated. Even worse, not all patients will benefit from the treatment, a proportion of them may suffer from severe side effects. By contrast, personalized medicine aims at treating patients differently with different drugs at the right dose [1]. To achieve personalized treatment for cancer, we need *biomarkers* (i.e. a set of genes) to predict

G. Livraga and S. Zhu (Eds.): DBSec 2017, LNCS 10359, pp. 185–204, 2017.
DOI: 10.1007/978-3-319-61176-1_10

a patient's response to anticancer drugs (e.g. sensitivity and resistance). With the advert of bioinformatics technology, we are able to make use of data mining and statistical methods to discover biomarkers from genomic data. Cancer Genome Project (CGP) [2] and Cancer Cell Line Encyclopedia (CCLE) [3] are examples showing the analysis results for discovering biomarkers using genomic features derived from human tumor samples against drug responses. A typical input for genomic features is a *gene expression profile* which is a vector recording the degrees of activation of different genes. The number of genes can be up to tens of thousands. A patient's response is usually measured by GI50 value (log of drug concentration for 50% growth inhibition) [4]. Given n gene expression profiles (e.g. from n patients) of which the dimension is m ($n \ll m$), the task is to perform regression analysis between gene expression profiles and GI50 values. Elastic net regression was found to be the most accurate predictor [5] among existing approaches.

Why encrypted by different keys? Due to the huge volume of medical records and DNA information, there is an increasing trend for medical units to make use of a third-party cloud system to store the records as well as to leverage its massive computational power to analyze the data. It is well recognized that genomic information such as DNA is particularly sensitive and must be well protected [6]. The privacy of gene expression has been overlooked until Schadt et al. pointed out that gene contents can be inferred based on expression profiles alone [7]. Even worse, some expression data is strongly correlated to important personal indexes such as body mass index and insulin levels. It is likely that an entire profile can be derived and linked to a specific individual. Therefore, gene expression profiles stored in cloud should also be encrypted. As medical units need to *retrieve* expression profiles when implementing personalized treatment, profiles from different medical units would be encrypted using *different keys* to avoid leaking the details of the records to other medical units.

It is important for different medical units to combine their datasets in order to increase the size of n (the number of patients) for accurate predication. Collaborative data mining on encrypted data is a promising direction for medical units to "share" data for more accurate prediction without jeopardizing the privacy of the data. The problem to be tackled in this paper is to design a privacy-preserving elastic net protocol to predict biomarkers based on gene expression profiles encrypted by different keys and GI50 values. Our goal is to ensure that the cloud learns nothing about the patients' expression profiles beyond what is revealed by the final result of elastic net regression.

Difficulties of the problem: There is no existing work for elastic net or lasso (another popular linear regression model) [8] while most of the work was designed for ordinary least square (OLS) and ridge regression. The difficulty lies on the fact that unlike solving OLS and ridge regression, the state-of-the-art solution (e.g. glmnet [9]) for elastic net is based on an iterative algorithm, which requires information of one training sample in each iteration. It is not clear how to perform these iterations if all data records are encrypted. Other existing solutions (e.g. Least Angle Regression (LARS) [10], computing the Euclidean projection onto a closed convex set [11] and using proximal stochastic dual coordinate descent [12]) suffer from a similar problem.

Ideas of our proposed solution: Instead of using the iterative algorithms to solve the elastic net problem directly, it has been proved that elastic net regression can be reduced to support vector machine (SVM) [13]. An identical solution as glmnet [9] up to a tolerance level can be obtained with a solver for SVM. Our main idea is to transform the encrypted training dataset of elastic net to that of SVM, based on which we compute the *gram matrix*[1]. Then the gram matrix will be used as input to a modern SVM solver. Once obtaining the solution to SVM, we reconstruct the solution to elastic net. We make sure that the cloud server cannot recover patients' expression profiles based on the gram matrix, for which we provide a security proof in this paper.

Roughly speaking, there are two ways to achieve privacy preserving SVM. One is perturbation based approach. Data sent to the cloud is perturbed by a random transformation [14], which considers only one user (i.e. medical unit). The other is cryptography based approach, such as secret sharing [15], Oblivious Transfer (OT) [15,16] and Fully Homomorphic Encryption (FHE) [15,17]. The cryptography based approach provides a higher level of privacy compared to perturbation based approach, but incurs higher computation/communication overhead. Most of the previous work focused on distributed databases [15,16,18–20], while we consider a centralized *outsourced encrypted database under multiple keys*. Liu et al. proposed a secure protocol based on FHE for outsourced encrypted SVM [17], but it requires the users to be online during the whole process. It has been proved that completely non-interactive multiple party computation cannot be achieved in the single server setting when user-server collusion might exist [21]. Thus, we need at least *two non-colluding servers* [22] if we want to keep the medical units offline. This two non-colluding servers model makes sense in the practical community (e.g. [22,23]). For example, we set up two cloud servers which belong to Amazon Web Services (AWS) cloud service and Google Cloud Platform (GCP) respectively. Considering the consequences of legal action for breach of contract and bad reputation, it is reasonable to assume that they will not collude. According to [24], each user can secret-share its data among the two non-colluding servers. Then the two servers compute on the shares of the input interactively and send the shares of the result to the users to reconstruct the final output. Although the secret sharing based approach is better in terms of computation cost, it incurs higher communication cost [25] and cannot deal with data encrypted under multiple keys. Moreover, oblivious transfer focuses on the single key setting, which is not suitable for the case of *multiple keys*. Consequently, we focus on homomorphic encryption based approach in this paper. There indeed exists a multikey FHE primitive that allows computation on data encrypted under multiple keys [26]. However, its efficiency is still far from practice and it requires interactions among all the medical units during the decryption phase. Peter et al. came up with a scheme that transforms the ciphertexts under different keys into those under the same

[1] A matrix that contains dot product of any two training samples.

key [27], incurring a huge amount of interactions between the servers. To reduce communication overhead, proxy re-encryption [30] can be utilized to transform ciphertexts [28,29]. However, the amount of interactions is still heavy. Because they used partially homomorphic encryption - if the underlying cryptosystem is additively homomorphic, they need joint work between the two servers to compute multiplication and vice versa. To further reduce the *communication overhead*, we utilize a framework to enable additively homomorphic encryption to support one multiplication [31]. We choose the BCP Cryptosystem [32] as the underlying additively homomorphic encryption and modify it to support multikey additive homomorphism. In this way, we successfully remove the need to transform the ciphertexts to those under the same key, while it is a must in [27–29]. To remove the constraint that medical units need to be online during decryption phase, we divide a medical unit's secret key s into two shares s_1 and s_2, and distribute them to the servers. Final decryption is obtained after two rounds of partial decryption.

To summarize, our contributions include the following:

(1) We construct a homomorphic cryptosystem that supports one multiply operation under single key and multiple add operations under both single key and different keys. Compared with the BCP cryptosystem, our scheme only doubles the encryption time. With 1024-bit security parameter, an add operation takes less than 1 ms while a multiply operation takes about 16 ms. The size of ciphertext increases linearly from 6138 B to 26 KB with the number of involved users increasing from 2 to 100. Overall speaking, the proposed scheme is practical.

(2) We propose the first privacy preserving protocol to solve elastic net on gene expression profiles encrypted by different encryption keys for cancer biomarker discovery, which encourages cooperation between medical units. Through reduction from elastic net to SVM, we demonstrate how to train SVM securely based on the gram matrix. The solution to elastic net is reconstructed based on the solution to SVM. Moreover, our solution can allow users (medical units) to stay offline except for the initialization phase.

(3) We evaluate our scheme on a real database[2] for drug sensitivity in cancer cell lines [33]. Moreover, our scheme can also be used to solve lasso, based on a similar reduction from lasso to SVM [34].

2 Model Description

In this paper, we propose a collaborative model for privacy preserving biomarker discovery for anticancer drugs using encrypted expression profiles extracted from the tumor samples of patients. As shown in Fig. 1, the involved parties are patients, medical units, certified institution and the cloud.

(1) Patients (P). Cancer patients go to the medical units to receive personalized treatment. We list six patients here labeled as $\{P_1, P_2 \cdots, P_6\}$.

[2] http://www.cancerrxgene.org, accessed on 10 Aug 2016.

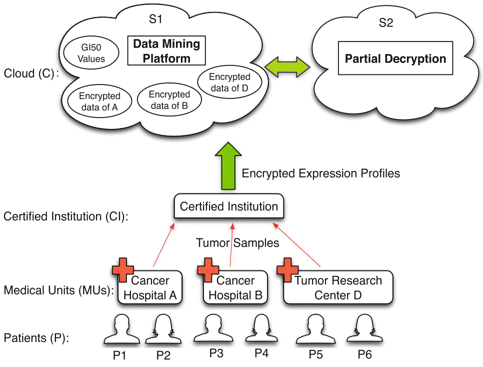

Fig. 1. System model for genomic biomarker discovery through collaborative data mining.

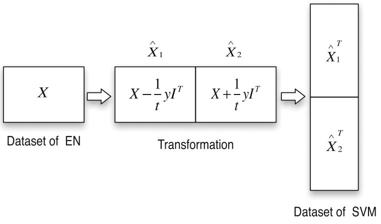

Fig. 2. Dataset transformation from elastic net to SVM

(2) Medical Units (MUs). There are different MUs (e.g. cancer hospitals, tumor research centers) in our model. Each MU is able to extract tumor samples from the patients, observe the effect of 72 h of anticancer drug treatment on them, and upload the GI50 values to the cloud. On the other hand, MU sends the tumor samples to the certified institution.

(3) Certified Institution (CI). CI is responsible to perform gene expression profiling. CI encrypts the gene expression profiles from different MUs with different encryption keys, and sends the encrypted profiles to the cloud. Only the MU that holds the correct private key can decrypt the encrypted profiles.

(4) Cloud (C). It consists of two non-colluding servers S_1 and S_2, which is responsible for storage and massive computation.

Threat model: CI is a trusted party. S_1 and S_2 are both honest-but-curious, and they are non-colluding. There might exist collusion between a MU and S_1. However, none of the medical units will collude with S_2. We consider two types of potential attacks: (i) attacker at one MU tries to know the expression profiles of other MUs (ii) attacker at S_1 or S_2 in the cloud aims at recovering gene expression profiles through observing the input, intermediate or final results.

3 Preliminaries

3.1 Elastic Net Regression

Let the input dataset be $\{(x_i, y_i)\}_{i=1}^{n}$, where each $x_i \in R^m$ is a column vector representing a gene expression profile, and $y_i \in R$ is the GI50 value.[3] Let $X \in R^{n\times m}$ be a matrix containing all gene expression profiles (the transposed

[3] GI50 denotes the log of the drug concentration for 50% growth inhibition.

i-th row of X is x_i) and the column vector $y \in R^n$ (i-th element of y is y_i) be the responses, the goal of linear regression analysis is to find a column vector $\beta \in R^m$ such that y_i can be approximated by $\tilde{y}_i = \beta^T x_i$. The ordinary least squares (OLS) regression works by minimizing the residual sum of squares $\min_\beta ||X\beta - y||_2^2$. There are some situations where OLS is not a good solution, for example, when m is large or the columns of X are highly correlated. One way to handle this problem is to introduce a penalization term. Ridge regression uses l_2-norm penalization ($||\beta||_2^2$), while lasso regression uses l_1-norm penalization ($|\beta|_1$). Ridge regression cannot produce a sparse model. By contrast, owing to the nature of l_1 penalty, lasso is able to generate a sparse model. Nevertheless, lasso has some limitations - it selects at most n variables in the $n \ll m$ case, picks out only one variable from a group of correlated variables not caring which one is selected (the robustness issue: we want to identify all related variables). In our application, since $n \ll m$ and genes may be highly correlated, lasso regression is not the ideal method in this situation and elastic net penalty ($\lambda_1|\beta|_1 + \lambda_2||\beta||_2^2$) is introduced, which is a convex combination of the lasso and ridge penalty [10]. It performs well under the situation of $n \ll m$ and correlated variables. The elastic net regression can be represented as follows.

$$\min_{\beta \in R^m} ||X\beta - y||_2^2 + \lambda||\beta||_2^2 \qquad \textit{such that } |\beta|_1 \leq t \tag{1}$$

where $\lambda > 0$ is the l_2-regularization constant and $t > 0$ is the l_1-norm budget.

3.2 Support Vector Machine with Squared Hinge Loss

Given that we have a dataset $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in R^m$ and $y_i \in \{+1, -1\}$, we aim at finding a separating plane $w^T x + b = 0$ ($w \in R^m$) to classify the training samples into two classes. There exists many eligible separating planes. For sake of robustness, support vector machine maximizes the margin ($\frac{1}{||w||}$) between two classes, which is equivalent to minimize $||w||^2$. However, sometimes the training dataset is linearly inseparable. One solution is to allow SVM to make mistakes on some samples. We use the squared hinge loss $max(0, 1 - y_i(w^T x_i + b))^2$ to measure the error of sample x_i, which need to be minimized. Therefore, the linear SVM with squared hinge loss can be represented as follows.

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^n max(0, 1 - y_i(w^T x_i + b))^2 \tag{2}$$

where C is the penalty parameter of the error term. The above is the primal form of SVM, which is often solved in its dual form:

$$\min_{\alpha_i \geq 0} \quad f(\alpha) = \alpha^T Q \alpha + \frac{1}{2C} \sum_{i=1}^n \alpha_i^2 - 2 \sum_{i=1}^n \alpha_i \tag{3}$$

where $\alpha \in R^n$ and each α_i is the coefficient for x_i. Q is a $n \times n$ matrix with $Q_{ij} = y_i y_j x_i^T x_j$. *Gram matrix K is defined as* $K = x_i^T x_j$. Once we get α by solving (3), we can further compute $w = \sum_{i=1}^n \alpha_i x_i y_i$.

3.3 Reduction from Elastic Net to SVM

Zhou et al. demonstrated that elastic net regression can be reduced to SVM [13]. They do not include any bias item b (they assume that the separating hyperplane passes through the origin). After a series of transformations, (1) and (3) can be changed to (4) and (5) respectively.[4] We do not provide the transformation steps (Please refer to [13] for details).

$$\min_{\hat{\beta}_i > 0} ||\hat{Z}\hat{\beta}||_2^2 + \lambda \sum_{i=1}^{2m} \hat{\beta}_i^2 \qquad \sum_{i=1}^{2m} \hat{\beta}_i = 1 \tag{4}$$

where $\hat{Z} = [\hat{X}_1, -\hat{X}_2] \in R^{n \times 2m}$, $\hat{X}_1 = X - \frac{1}{t} y I^T$ and $\hat{X}_2 = X + \frac{1}{t} y I^T$ ($I \in R^m$ is an identity vector).

$$\min_{\alpha_i > 0} ||Z(\frac{\alpha}{|\alpha^*|_1})||_2^2 + \lambda \sum_{i=1}^{2m} (\frac{\alpha_i}{|\alpha^*|_1})^2 \qquad \sum_{i=1}^{2m} \frac{\alpha_i}{|\alpha^*|_1} = 1 \tag{5}$$

where $Z = y_i x_i$, $C = \frac{1}{2\lambda}$ and α^* is the optimal solution. Comparing (4) and (5), we notice that they have similar form except for two differences. The first one is that the class labels in elastic net are real valued but binary in SVM. As shown in Fig. 2, to transform the training dataset X of elastic net to that of SVM, we compute $\hat{X}_1$ as subtracting each column of X by $\frac{1}{t} y$ and calculate $\hat{X}_2$ as adding each column of X by $\frac{1}{t} y$, then concatenate $\hat{X}_1$ and $\hat{X}_2$ together and transpose it. The first m training samples of SVM are of class $+1$, and the remaining are of class -1. The second difference is that they have different scale. The optimal solution $\hat{\beta}^*$ can be represented by the optimal solution α^* as $\hat{\beta}^* = \frac{\alpha^*}{|\alpha^*|_1}$. Finally, the optimal solution β to elastic net (see (1)) can be recovered from $\hat{\beta}$ according to $\beta = t \times (\hat{\beta}_{1 \cdots m} - \hat{\beta}_{m+1 \cdots 2m})$, where t is the l_1-norm budget and $\hat{\beta}_{i \cdots j}$ denotes a vector consisting of elements of $\hat{\beta}$ from index i to j.

4 Our Scheme

Fully homomorphic encryption can be used to compute arbitrary polynomial functions over encrypted data. However, the high computational complexity and communication cost preclude its use in practice. If only focusing on those operations of interest to the target application, more practical homomorphic encryption schemes are possible. For example, Zhou and Wornell [35] proposed an integer vector encryption scheme which supports addition, linear transformation and weighted inner product on ciphertexts. Nevertheless, reduction from elastic net to SVM leads to changes of the training dataset. To be specific, one gene expression profile of a patient across all genes (i.e. one row) is a training sample of elastic net. But one training sample of SVM see Fig. 2 can be considered as gene expression values of a particular gene among all the patients (i.e. one column).

[4] Here we use $\hat{\beta} \in R^{2m}$ to differentiate from $\beta \in R^m$, β can be derived from $\hat{\beta}$.

Therefore, if we encrypt gene expression profiles using the integer vector encryption, there is no way to construct ciphertexts for the training dataset of SVM. As a result, we restrict our attention to cryptosystems encrypting one element of the profile at a time instead of encrypting the whole profile. Recall that we can use gram matrix as input to train SVM (see Sect. 3.2) and the basic operation to compute gram matrix is the dot product of two samples, it requires a ciphertext to support one multiply operation and multiple add operations. Indeed, the BGN cryptosystem [36] can compute one multiplication on ciphertexts using the bilinear maps. However, it does not support multikey homomorphism. In our setting of collaborative data mining in the cloud, the training dataset of elastic net is horizontally partitioned (different units holding different records with the same set of attributes) while the training dataset of SVM is vertically partitioned (records are partitioned into different parts with different attributes after the transformation). In order to train SVM on encrypted training dataset, we thus need a cryptosystem that *supports one multiply operation under single key and multiple add operations under both single key and different keys.* In this paper, we try to let medical units stay offline except for the initialization phase. Specifically, we use secret sharing to authorize one server (i.e. S_1) to decrypt the encrypted gram matrix without knowing the secret key of any medical unit.

4.1 Building Blocks

Framework to Enable One Multiplication on Cihphertexts. Catalano and Fiore [31] showed a framework to enable existing additively homomorphic encryption schemes (i.e. Paillier, ElGamal) to compute multiplication on ciphertexts. We use $E(\,)$ to denote the underlying additively homomorphic encryption. The idea is to transform a ciphertext $E(x_{ij})$ into "multiplication friendly". To be specific, we use $\mathcal{E}(x_{ij}) = (x_{ij} - b_{ij}, E(b_{ij}))$ (where b_{ij} is a random number) to represent the "multiplication friendly" ciphertext. Given two "multiplication friendly" ciphertexts $\mathcal{E}(x_{11}) = (x_{11} - b_{11}, E(b_{11}))$ and $\mathcal{E}(x_{21}) = (x_{21} - b_{21}, E(b_{21}))$, we compute multiplication as $\mathcal{E}(x_{11}x_{21}) = (\alpha_1, \beta_1, \beta_2)$.

$$\begin{aligned} \alpha_1 &= E[(x_{11} - b_{11})(x_{21} - b_{21})]E(b_{11})^{x_{21}-b_{21}}E(b_{21})^{x_{11}-b_{11}} \\ &= E(x_{11}x_{21} - b_{11}b_{21}) \qquad (6) \end{aligned}$$

$$\beta_1 = E(b_{11}) \qquad \beta_2 = E(b_{21}) \qquad (7)$$

To decrypt $\mathcal{E}(x_{11}x_{21})$, we will add $b_{11}b_{21}$ to the decryption of α where b_{11}, b_{21} is retrieved from β_1, β_2. The addition of two ciphertexts after multiplication works by adding the α components and concatenating the β components. Therefore, the β component will grow linearly with additions after performing a multiplication. To remove this constraint, two non-colluding servers are used to store $\mathcal{E} = (x_{ij} - b_{ij}, E(b_{ij}))$ and b_{ij} respectively. In this way, S_1 can throw away the β component after performing a multiplication, because S_2 will operate on the b_{ij}'s in plaintext. Therefore, the ciphertext contains only the α component after performing a multiplication. This framework has a nice property that it *inherits the multikey homomorphism* of the underlying additively homomorphic encryption.

Given two safe primes p and q, we compute $N = pq$, $g = -a^{2N}$ $(a \in \mathbb{Z}^*_{N^2})$, the secret key $s \in [1, \frac{N^2}{2}]$, the public key $(N, g, h = g^s)$.

Encryption: To encrypt plaintext $m \in \mathbb{Z}_N$, we select a random $r \in [1, \frac{N}{4}]$ and generate the ciphertext $E(m) = (C^{(1)}_m, C^{(2)}_m)$ as below:

$$C^{(1)}_m = g^r \, mod \; N^2 \quad and \quad C^{(2)}_m = h^r(1 + mN) \; mod \; N^2 \tag{8}$$

Decryption:

$$t = \frac{C^{(2)}_m}{(C^{(1)}_m)^s} \qquad m = \frac{t - 1 \; mod \; N^2}{N} \tag{9}$$

Proxy Re-encryption: If the secret key s is divided into two shares s_1, s_2 such that $s = s_1 + s_2$, then we can use s_1 to partially decrypt $E(m)$ to $E(m)' = (C^{(1)}_m{}', C^{(2)}_m{}')$, which can be considered as a ciphertext under key s_2.

$$C^{(1)}_m{}' = C^{(1)}_m \qquad C^{(2)}_m{}' = \frac{C^{(2)}_m}{C^{(1)}_m{}^{s_1}} \tag{10}$$

Single Key Homomorphism: Supposed that we have two plaintexts m_1, m_2 and their ciphertexts $E(m_1) = (C^{(1)}_{m_1}, C^{(2)}_{m_1})$ and $E(m_2) = (C^{(1)}_{m_2}, C^{(2)}_{m_2})$ under the same key s. The ciphertext $E(m_1 + m_2)$ can be computed as $E(m_1 + m_2) = (C^{(1)}_{m_1} C^{(1)}_{m_2}, C^{(2)}_{m_1} C^{(2)}_{m_2})$.

Fig. 3. The BCP cryptosystem

Multikey Homomorphism of the BCP Cryptosystem. The BCP cryptosystem (also known as Modified Paillier Cryptosystem) is an additively homomorphic encryption under single key [32]. We briefly review the BCP cryptosystem in Fig. 3 and discuss how to modify it to support multikey homomorphism at the expense of expanding the ciphertext size. Supposed that $E(m_a) = (C^{(1)}_{m_a}, C^{(2)}_{m_a})$ is under key s_a, $E(m_b) = (C^{(1)}_{m_b}, C^{(2)}_{m_b})$ is under key s_b, then $E^{(ab)}(m_a + m_b)$ where $E^{(ab)}$ denotes a ciphertext related to key s_a and s_b can be computed as

$$E^{(ab)}(m_a + m_b) = (C^{(1)}_{m_a}, C^{(1)}_{m_b}, C^{(2)}_{m_a} C^{(2)}_{m_b}) \tag{11}$$

The ciphertext size *only depends on the number of involved medical units (i.e. keys)*. There are two MUs with key s_a and s_b respectively in this example, the addition of their ciphertexts is a 3-tuple. If n MUs cooperate together, the addition of their ciphertexts should be a $(n+1)$-tuple. To decrypt $E^{(ab)}(m_a + m_b)$, the secret key s_a and s_b are required.

$$t = \frac{C^{(2)}_{m_a} C^{(2)}_{m_b}}{(C^{(1)}_{m_a})^{s_a} (C^{(1)}_{m_b})^{s_b}} \qquad m_a + m_b = \frac{t - 1 \; mod \; N^2}{N} \tag{12}$$

Incorporating the above modified the BCP cryptosystem to the framework that enables additively homomorphic encryption to support one multiplication, we obtain our final encryption scheme $\mathcal{E}_{BCP}$.[5]

[5] $\mathcal{E}$ denotes the framwork and BCP denotes the underlying cryptosystem.

Gram Matrix Computation. Gram matrix K is defined as $K_{ij} = \langle x_i, x_j \rangle = x_i^T x_j$ where x_i and x_j are any two training samples (see Sect. 3.2). Recall that the original training dataset X of elastic net regression is transformed to the training dataset $\hat{X}$ of SVM during the reduction process (see Sect. 3.3), we use $\hat{X} = \{\hat{x}_i\}_{i=1}^{2m}$ to denote the transformed dataset. After dataset transformation, the horizontally partitioned dataset of the elastic net is converted to vertically partitioned dataset of SVM. The gram matrix $K(\hat{X})$ of $\hat{X}$ is computed as follows.

$$K(\hat{X}) = \begin{bmatrix} \langle \hat{x}_1, \hat{x}_1 \rangle & \langle \hat{x}_1, \hat{x}_2 \rangle & \cdots & \langle \hat{x}_1, \hat{x}_{2m} \rangle \\ \langle \hat{x}_2, \hat{x}_1 \rangle & \langle \hat{x}_2, \hat{x}_2 \rangle & \cdots & \langle \hat{x}_2, \hat{x}_{2m} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \hat{x}_{2m}, \hat{x}_1 \rangle & \langle \hat{x}_{2m}, \hat{x}_2 \rangle & \cdots & \langle \hat{x}_{2m}, \hat{x}_{2m} \rangle \end{bmatrix} \tag{13}$$

For ease of description, we firstly consider the case of two medical units denoted as MU$_A$ and MU$_B$. Assume that the cloud store n gene expression profiles, among which n_A records are from MU$_A$ and n_B records are from MU$_B$. Then in the transformed dataset of SVM, for each training sample, the first n_A elements are encrypted under key s_A, the remaining n_B elements are encrypted under key s_B. Assume that we have two training samples $\hat{x_1}$ and $\hat{x_2}$ of SVM, their dot product $\langle \hat{x}_1, \hat{x}_2 \rangle$ can be computed as follows and their ciphertexts are denoted as $\mathcal{E}_{BCP}(\hat{x}_1) = (\hat{x}_1 - b_1, E(b_1))$, $\mathcal{E}_{BCP}(\hat{x}_2) = (\hat{x}_2 - b_2, E(b_2))$.

$$\langle \hat{x}_1, \hat{x}_2 \rangle = \sum_{i=1}^{n_A} \hat{x}_{1i}\hat{x}_{2i} + \sum_{i=n_A+1}^{n} \hat{x}_{1i}\hat{x}_{2i} \tag{14}$$

Supposed that the ciphertext of $\sum_{i=1}^{n_A} \hat{x}_{1i}\hat{x}_{2i}$ and $\sum_{i=n_A+1}^{n} \hat{x}_{1i}\hat{x}_{2i}$ are α_A and α_B respectively,[6] then S_1 will compute α_A, α_B as follows. The computation of α_A or α_B only requires single key homomorphism.

$$\alpha_A = E(\sum_{i=1}^{n_A} \hat{x}_{1i}\hat{x}_{2i} - b_{1i}b_{2i}) = (C_A^{(1)}, C_A^{(2)}) \tag{15}$$

$$\alpha_B = E(\sum_{i=n_A+1}^{n_A+n_B} \hat{x}_{1i}\hat{x}_{2i} - b_{1i}b_{2i}) = (C_B^{(1)}, C_B^{(2)}) \tag{16}$$

As α_A and α_B are encrypted under different keys, adding them together requires multikey homomorphism.

$$E(\langle \hat{x}_1, \hat{x}_2 \rangle - \langle \hat{b}_1, \hat{b}_2 \rangle) = \alpha_A + \alpha_B = (C_A^{(1)}, C_B^{(1)}, C_A^{(2)} C_B^{(2)}) \tag{17}$$

Keep Medical Units Offline. We leverage $\mathcal{E}_{BCP}$'s proxy re-encryption property, which inherits from the underlying the BCP cryptosystem (see (10)). To keep MU$_A$ and MU$_B$ offline, we split the secret key s of each involved medical

[6] S_1 will abandon the β_1 and β_2 component after a multiplication.

unit into two shares. Specifically, we have $s_A = s_{A_1} + s_{A_2}$ and $s_B = s_{B_1} + s_{B_2}$. S_1 holds s_{A_1}, s_{B_1} and S_2 holds s_{A_2}, s_{B_2}. To compute $\langle \hat{x}_1, \hat{x}_2 \rangle$, S_1 will firstly decrypt (17) partially.

$$C_A^{(1)'} = C_A^{(1)} \qquad C_B^{(1)'} = C_B^{(1)} \tag{18}$$

$$C_A^{(2)'} C_B^{(2)'} = \frac{C_A^{(2)} C_B^{(2)}}{(C_A^{(1)})^{s_{A_1}} (C_B^{(1)})^{s_{B_1}}} \tag{19}$$

Then S_1 will send $C_A^{(1)}$ and $C_B^{(1)}$ to S_2. S_2 will compute and return $(C_A^{(1)})^{s_{A_2}}$, $(C_B^{(1)})^{s_{B_2}}$, $\langle \hat{b}_1, \hat{b}_2 \rangle$ to S_1 afterwards. Finally, S_1 is able to decrypt $E(\langle \hat{x}_1, \hat{x}_2 \rangle - \langle \hat{b}_1, \hat{b}_2 \rangle)$ completely and get $\langle \hat{x}_1, \hat{x}_2 \rangle$ in plaintext.

$$C_A^{(2)''} C_B^{(2)''} = \frac{C_A^{(2)'} C_B^{(2)'}}{(C_A^{(1)'})^{s_{A_2}} (C_B^{(1)'})^{s_{B_2}}} \tag{20}$$

$$\langle \hat{x}_1, \hat{x}_2 \rangle = \frac{(C_A^{(2)''} C_B^{(2)''} - 1) \ mod \ n^2}{n} + \langle \hat{b}_1, \hat{b}_2 \rangle \tag{21}$$

The above shows how to compute $\langle \hat{x}_1, \hat{x}_2 \rangle$ based on two ciphertexts $\mathcal{E}_{BCP}(\hat{x}_1)$ and $\mathcal{E}_{BCP}(\hat{x}_2)$. Similarly, we can compute each element of the gram matrix $K_{ij} = \langle \hat{x}_i, \hat{x}_j \rangle = \hat{x}_i^T \hat{x}_j$ based on the ciphertexts $\mathcal{E}_{BCP}(\hat{x}_i)$ and $\mathcal{E}_{BCP}(\hat{x}_j)$. Observing that the gram matrix in (13) is symmetric, we can only compute the upper triangular half of it. In the end, S_1 gets the gram matrix K in plaintext. If there are more than two medical units, we can easily extend (14), (15) and (17) to handle the case of multiple medical units. The size of ciphertext of $\langle \hat{x}_1, \hat{x}_2 \rangle$ increases linearly with the number of involved medical units. Likewise, the communication overhead also increases linearly during the decryption phase.

4.2 Our Construction

Given the encrypted gene expression profiles $\mathcal{E}_{BCP}(X)$ derived from multiple medical units, the cloud runs privacy preserving elastic net on it to discover biomarkers to predict a patient's response to anticancer drugs. As it is not clear how to design a privacy preserving protocol based on iterative algorithms to solve elastic net. We resort to reduction to shift our attention from elastic net to SVM. In Algorithm 1, we firstly demonstrate how to transform the encrypted dataset of elastic net to that of SVM (see Sect. 3.3). It is easy to perform such transformation on the dataset in plaintext. However, once it is encrypted, we need to rely on the homomorphic properties of our cryptosystem to finish the transformation. Next, we compute the encrypted gram matrix $\mathcal{E}_{BCP}(K)$ of the transformed training dataset (see Sect. 4.1). Gram matrix plays a role as intermediate dataset based on which SVM model can be generated correctly without breaching the privacy of patients' gene expression profiles. In order to keep medical units offline, we authorize S_1 to decrypt $\mathcal{E}_{BCP}(K)$. Based on K, we train SVM and obtain the solution α. Finally, we use α to reconstruct β, which is the solution to elastic net.

Algorithm 1. Protocol for Privacy-preserving Elastic Net.

Input: Encrypted dataset $\mathcal{E}_{BCP}(X) = \{\mathcal{E}_{BCP}(x_i)\}_{i=1}^{n}$, where $x_i \in Z^m$ and response vector in plaintext $y \in R^n$; l_1-norm budget t and l_2-regularization parameter λ.
Output: Solution β.

1. Dataset Transformation: Compute $\mathcal{E}_{BCP}(\hat{X}_1^T)$ and $\mathcal{E}_{BCP}(\hat{X}_2^T)$, where $\hat{X}_1 = X - \frac{1}{t}yI^T$ and $\hat{X}_2 = X + \frac{1}{t}yI^T$. Given t and y, we might need to scale $\mathcal{E}_{BCP}(X)$ to make sure operations are run on integer domain. The encrypted training dataset of SVM is $\mathcal{E}_{BCP}(\hat{X}) = [\mathcal{E}_{BCP}(\hat{X}_1), \mathcal{E}_{BCP}(\hat{X}_2)]^T$ and the class labels $\hat{y} \in Z^{2m}$ where $\hat{y}_i = +1$ if $i \in [1, m]$ and $y_i = -1$ if $i \in [m+1, 2m]$.
2. Gram Matrix Computation: Compute first the encrypted gram matrix $\mathcal{E}(K)$ of SVM based on $\mathcal{E}_{BCP}(\hat{X})$, then authorize S_1 to decrypt $\mathcal{E}(K)$ and get the gram matrix K in clear.
3. Train SVM: Solve the dual optimization problem of SVM (see (3)) based on gram matrix K and $C = \frac{1}{2\lambda}$ to get SVM's solution α (Please refer to the Appendix if readers are interested in how to train SVM).
4. Reconstruct elastic net's solution β based on α.

Model Assessment. In Algorithm 1, there are two parameters: l_1-norm constraint t and l_2-regularization parameter λ. It is not known beforehand which t and λ are best for the elastic net. For different combinations of (t, λ), the predictive power of the derived solution varies. We do "grid search" on t and λ using k-fold cross validation [37] to assess the goodness-of-fit of our model under different parameters. The grid-search is straightforward. We specify the range of t and λ respectively. Then we try various pairs of (t, λ). As it might be time-consuming to do a complete grid-search, we recommend using a coarse grid first. Once a "better" region is identified, we will conduct a finer grid search on that region. We divide our training dataset $\mathcal{E}_{BCP}(X)$ into k subsets satisfying $\mathcal{E}_{BCP}(X) = \mathcal{E}_{BCP}(X_1) \cup \cdots \cup \mathcal{E}_{BCP}(X_k)$, $\mathcal{E}_{BCP}(X_i) \cap \mathcal{E}_{BCP}(X_j) = \emptyset$ $(i \neq j)$. We use $\mathcal{E}_{BCP}(X_i)$ where $i \in [1, k]$ as the validation set and the remaining $k-1$ subsets as the training dataset each time. In order to measure the performance of regression, we choose Rooted Mean Squared Error (RMSE). An RMSE value closer to 0 indicates the regression model is more useful for prediction. In the setting of k-fold cross validation, we need to compute the average of k RMSE values. Supposed that there are d samples in the validation set, the predicted GI50 value of gene expression profile x_i is $\tilde{y}_i$ and the true value is y_i, then RMSE is computed as

$$RMSE = \sqrt{(\frac{1}{d}\sum_{i=1}^{d}(\tilde{y}_i - y_i)^2)} \tag{22}$$

Recall that each gene expression profile is encrypted, we can compute the ciphertext of the predicted GI50 value $\tilde{y}_i$ as $\mathcal{E}_{BCP}(\tilde{y}_i) = (\beta^T(x_i - b_i), \beta^T E(b_i))$ where β is the solution to elastic net. To get $\tilde{y}_i$ in plaintext, we make the two non-colluding servers work together. S_1 reveals β to S_2. S_2 return $\beta^T b_i$ to S_1. Then S_1 computes $\tilde{y}_i = \beta^T(x_i - b_i) + \beta^T b_i = \beta^T x_i$. For each (t, λ) pair, S_1 computes RMSE with each predicted value $\tilde{y}_i$. Finally, S_1 will pick the optimal (t, λ) which achieves the smallest RMSE and get the optimal solution β^*.

5 Security Analysis

We consider the honest-but-curious model, meaning that all the medical units, S_1 or S_2 will follow our protocol but try to gather information about the inputs of MUs. There might exist collusion between a medical unit and S_1. We analyze the security of our model with the Real and Ideal paradigm and Composition Theorem [38]. The main idea is to use a simulator in the ideal world to simulate the view of a semi-honest adversary in the real world. If the view in the real world is computationally indistinguishable from the view in the ideal world, then the protocol is believed to be secure. According to the Composition Theorem, the entire scheme is secure if each step is proved to be secure. Due to page limit, the Proof of Theorem 1 is given in the Appendix.

Theorem 1. *In Algorithm 1, it is computationally infeasible for S_1 to distinguish the gene expression profiles encrypted under multiple keys as long as $\mathcal{E}_{BCP}$ is semantically secure and the two servers are non-colluding.*

Theorem 2. *No encryption scheme is secure against known-sample attack if dot products are revealed.*

Proof: We define known-sample attack as an attacker obtaining the plaintexts of a set of records of the encrypted database but not knowing the correspondence between the plaintexts and the encrypted records. According to [39], no encryption scheme is secure against known-sample attack if distance information is revealed. As distance computation can be decomposed into dot products, revealing dot products equals to revealing distance. Given n encrypted samples whose dimension is m, if an attacker knows the plaintexts of m linearly independent samples, the attacker can obtain the plaintext of any encrypted samples even without the decryption key. The idea is to construct m linear equations, whose unique solution corresponds to the desired sample.

Fortunately, in the following theorem, we show that it is impossible for the attackers to make use of Theorem 2 to launch the attack.

Theorem 3. *S_1 cannot reconstruct gene expression profile of a patient with gram matrix K known, considering the impossibility that an attacker collects enough samples of SVM to launch the attack mentioned in Theorem 2.*

Proof: According to Sect. 3.3, one gene expression profile of a patient across all genes (i.e. one row) is a training sample of elastic net. But the training sample of SVM can be considered as gene expression values of a particular gene among all the patients (i.e. one column). If S_1 colludes with MU_1, it only brings minor advantage that some of the elements from MU_1 of a training sample are revealed. Unless the attacker cracks our cryptosystem and obtains all the private keys of the involved medical units, he cannot set up linear equations to launch known-sample attack.

6 Experimental Evaluation

The configuration of our PC is Windows 7 Enterprise 64-bit Operating System with Intel(R) Core(TM) i5 CPU (4 cores), 3.4 GHz and 16 GB memory. We use a public database for drug sensitivity in cancer cell lines [33]. To provide platform independence, we use Java to implement our scheme together with open-source IDE. We use BigInteger class to process big numbers which offers all basic operations we need. We utilize SecureRandom class to produce a cryptographically strong random number. As for the generation of safe prime numbers, we use the probablePrime method provided by BigInteger class. The probability that a BigInteger returned by this method is composite does not exceed 2^{-100}. The performance of our scheme depends heavily on the size of modulus N, and the number of additions and multiplications performed. During the initialization phase, public and private key pair are generated. The runtime of generating a key pair varies with the bit length of N, as it depends a lot on the random number generator. A typical value for N is 1024 and it takes about 2 s in average to generate one key pair. We firstly compare the encryption time of two training samples when using the BCP Cryptosystem and our proposed cryptosystem $\mathcal{E}_{BCP}$. We vary the dimension of each sample from 1000 to 10000. The bit length of modulus N is set to 1024 and 1536 respectively (following the same setting as in [27]). As shown in Fig. 4, the encryption time scales linearly as the dimension increases. We use $E(x)$ and $(x - b, E(b))$ where b is a random number to denote the ciphertext of x under BCP and $\mathcal{E}_{BCP}$ cryptosystem respectively. Leveraging the framework proposed in [31], the encryption time of $\mathcal{E}_{BCP}$ doubles that of the BCP Cryptosystem. The additional encryption time is caused by generating the random number b. Moreover, we measure the time to compute dot product of two encrypted training samples. We focus on vertically partitioned dataset of SVM. To facilitate understanding, we encrypt the first half (belonging to Alice) of a sample using secret key s_A and the second half (belonging to Bob) using secret key s_B. We show the runtime of dot product computation on the ciphertexts in Fig. 5. Similarly, time to calculate dot product increases linearly with the dimension of samples. For vectors of dimension m, one dot product operation includes m multiplications and $m - 1$ additions, among which one addition is multikey homomorphic. It takes only 1 ms to run a multikey homomorphic addtition. For operations under single key, addition is much faster than multiplication. With a 1024-bit modulus, the runtime of additions is less than 1 s. The runtime of multiplications varies from 16 s to 185 s with the dimension of a sample increasing from 1000 to 10000. Therefore, multiplications are the bottleneck of dot product

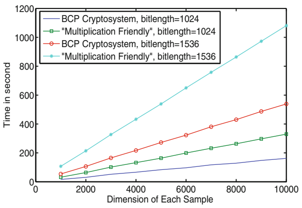

Fig. 4. Encryption time

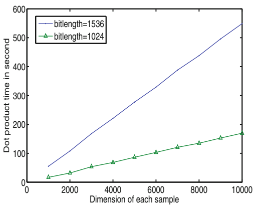

Fig. 5. Dot product time

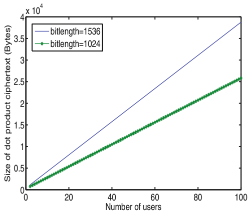

Fig. 6. Ciphertext size

computation. To decrypt one encryted dot product, it takes 285 ms and 572 ms with and without secret sharing separately. Recall that the multikey homomorphism property is achieved at the expense of expanding ciphertext size, we also measure the effect of the number of involved users on the increase of ciphertext size. As shown in Fig. 6, the ciphertext size increases linearly from 6138 B to 26 KB when the number of involved users increasing from 2 to 100.

The public database for drug sensitivity in this paper consists of 1002 cancer cell lines, 265 anticancer drugs. For each drug, GI50 values of around 300 to 1000 cell lines are available. As for gene expression profiling, it contains the RMA-normalized expression values of 17737 genes of 1018 cell lines. We preprocess them in MATLAB, keeping those cell lines that belong to the intersection of gene expression profiles and GI50 values. For example, considering drug *PD*-0325901, we get 843 expression profiles and GI50 values. As our cryptosystem only supports operations on the integer domain, we need to preprocess the database. To be specific, we first select a system parameter p to represent the number of bits for the fractional part of expression values. We next multiply each expression value by 10^p to get its integer value. Then, we need to divide each element of the gram matrix by 10^{2p} to remove the influence of scaling up. After running our privacy-preserving elastic net, we successfully pick out 165 genomic biomarkers.

Comparison with existing schemes: We focus on the homomorphic encryption based schemes [17, 27–29], of which the setting is outsourced encrypted database under multiple keys. According to the experimental evaluation of [31], using their proposed framework to enable one multiplication on additively homomorphic ciphertexts outperforms the BGV homomorphic encryption [40] (in terms of ciphertext size, time of encryption/decryption/homomorphic operations). As shown in the experiments above, modification to the BCP Cryptosystem for multikey homomorphism only doubles the encryption time. Therefore, addition and multiplication can be run more efficiently in our scheme compared to [17], which deals with only two users (i.e. keys). Besides, they require the users to be online while we keep the users offline in this paper. Under two non-colluding servers model, the schemes in [27–29] can be used to compute addition/multiplication. The main drawback of their schemes is that they have to transform the ciphertexts under multiple keys to those under the same key, which is a heavy workload for the cloud server. Moreover, computing multiplication incurs interactions between the two servers. By contrast, our cryptosystem enables calculating multiplication without interactions.

7 Discussion and Conclusions

In practical scenarios, a gene expression profile typically has dimension of order 10^4. We can only collect hundreds of patients' profiles of different cancers. If we store gram matrix K in the memory, it is of order 10^8 in our case, which requires a lot of memory. Keerthi et al. [41] proposed to restrict the support vectors to some subset of *basis vectors* $\mathcal{J} \subset \{1, \cdots, n\}$ in order to reduce the memory requirement. This method requires $O(|\mathcal{J}|n)$ space where $|\mathcal{J}| \ll n$. However, the derived α using this method can be different from the one we get using the entire gram matrix K. It is a trade-off between accuracy and efficiency. For genomic biomarker discovery, it is obviously more important to pick out accurate biomarkers. Therefore, it makes sense to maintain the gram matrix in memory. Furthermore, each element of the gram matrix can be calculated independently. To accelerate the computation of gram matrix, we can utilize existing parallel computing frameworks.

To conclude, in this paper, by assuming the existence of two non-colluding servers, we proposed a privacy preserving collaborative model to conduct elastic net regression through reduction to SVM on encrypted gene expression profiles and GI50 values of anticancer drugs. To compute the gram matrix on ciphertexts, we successfully construct a cryptosystem that supports one multiply operation under single key and multiple add operations under both single key and different keys. Besides, we use secret sharing to allow one of the cloud server to get the gram matrix. Our scheme keeps the medical units offline and is proved to be secure in the semi-honest model or even if a medical unit colludes with one cloud server. The experimental results highlight the practicability of our scheme. The proposed protocol could also be applied to other applications that use elastic net or lasso for linear regression. Our future work is to extend our scheme to malicious adversaries (either S_1 or S_2 is malicious). One promising direction is to use commitment scheme [42] and zero-knowledge protocols.

Acknowledgements. This work is supported in part by RGC CRF (Project No. CityU C1008-16), Hong Kong and National High Technology Research and Development Program of China (No. 2015AA016008).

Appendix

Train SVM: We do not include any bias item in this paper, according to Sect. 3.3. Therefore, efficient dual coordinate descent method [43] can be used, of which the main idea is to optimize one variable once at a time, reducing memory requirements. However, coordinate descent methods are inherently sequential and hard to parallelize. To utilize the parallel properties of SVM, we also seek to find an solution which can be parallelized [44]. As shown in [45], optimizing on either the primal problem or the dual problem is in fact equivalent. Linear SVM can be considered as non-linear SVM with a linear kernel k and an associated

Reproducing Hilbert Space $\mathcal{H}$.

$$\min_{f \in \mathcal{H}} \frac{1}{2}||f||^2_{\mathcal{H}} + C \sum_{i=1}^{n} max(0, 1 - y_i f(x_i))^2 \tag{23}$$

According to the *representer theorem*, the *minimizer* of (23) can be represented by $f^*(x) = \sum_{i=1}^{n} \theta_i k(x_i, x)$. Note that these coefficients θ_i are different from the Lagrange multipliers α_i in standard SVM literature. The relationship between θ_i and α_i is $\theta_i = y_i \alpha_i$. (23) can be rewritten as

$$\min_{\theta} \frac{1}{2}\theta^T K \theta + C \sum_{i=1}^{n} max(0, 1 - y_i \theta^T K_i)^2 \tag{24}$$

where K is also the gram matrix with $K_{ij} = x_i^T x_j$ and K_i is the i_{th} row of K. The Newton optimization algorithm of the above problem can be expressed as dense linear algebra operations. When combined with highly optimized libraries such as Intel's MKL for multicores, Jacket, and CuBLAS for GPUs, we can largely speedup the training of SVM.

Proof of Theorem 1. We discuss the security of each step in Algorithm 1.

Step 1 Dataset Transformation: Given $\mathcal{E}_{BCP}(X)$, it requires homomorphic addition to compute $\mathcal{E}_{BCP}(\hat{X}_1)$ and $\mathcal{E}_{BCP}(\hat{X}_2)$. Therefore, we need to prove the security of addition over ciphertexts against a semi-honest adversary $\mathcal{A}^{\mathbb{SH}}_{S_1}$ in the real world. We set up a simulator $\mathcal{F}^{\mathbb{SH}}$ in the ideal world to simulate the view of $\mathcal{A}^{\mathbb{SH}}_{S_1}$. Considering one operation $\mathcal{E}_{BCP}(X_{ij} + \frac{1}{t}y)$, The view of $\mathcal{A}^{\mathbb{SH}}_{S_1}$ in this step includes input $\{\mathcal{E}_{BCP}(X_{ij}), \mathcal{E}_{BCP}(\frac{1}{t}y)\}$ and output $\mathcal{E}_{BCP}(X_{ij} + \frac{1}{t}y)$. Without loss of generality, we assume that simulator $\mathcal{F}^{\mathbb{SH}}$ computes $\mathcal{E}_{BCP}(m_1)$ and $\mathcal{E}_{BCP}(m_2)$ where $m_1 = 1$ and $m_2 = 2$. Then the simulator computes $\mathcal{E}_{BCP}(m_1 + m_2)$ and returns $\{\mathcal{E}_{BCP}(m_1), \mathcal{E}_{BCP}(m_2), \mathcal{E}_{BCP}(m_1 + m_2)\}$ to $\mathcal{A}^{\mathbb{SH}}_{S_1}$. Since the view of $\mathcal{A}^{\mathbb{SH}}_{S_1}$ are ciphertexts generated under $\mathcal{E}_{BCP}$ cryptosystem and $\mathcal{A}^{\mathbb{SH}}_{S_1}$ has no knowledge of the private key. If $\mathcal{A}^{\mathbb{SH}}_{S_1}$ could distinguish the real world from the ideal world, then it indicates $\mathcal{A}^{\mathbb{SH}}_{S_1}$ is able to distinguish ciphertexts generated by $\mathcal{E}_{BCP}$, which contradicts to the assumption that $\mathcal{E}_{BCP}$ is semantically secure. Therefore, $\mathcal{A}^{\mathbb{SH}}_{S_1}$ is computationally infeasible to distinguish the real world from the ideal world.

For the case where a medical unit (denoted as MU_1) colludes with S_1, we use $\mathcal{A}^{\mathbb{SH}}_{(S_1, MU_1)}$ to denote the corresponding adversary. $\mathcal{A}^{\mathbb{SH}}_{(S_1, MU_1)}$ cannot learn anything beyond gene expression values of MU_1.

Step 2 Gram Matrix Computation: Recall that the basic operation of gram matrix computation is dot product of two training samples $\hat{x}_1$ and $\hat{x}_2$. The security of addition has been proved in step 1. As for the security of multiplication, computing α component is implemented over ciphertexts on S_1 (see (6)). Similar to the proof above, we can prove the security of multiplication with the real and ideal paradigm. Moreover, the multikey homomorphic addition of the BCP

Cryptosystem is based on ciphertexts (see (11)). As long as the BCP Cryptosystem is semantically secure, the multikey homomorphic addition is secure. As for decrypting the encrypted dot product, S_1 interacts with S_2. S_1 sends C_{A_1} and C_{B_1} to S_2 (see (18)). S_2 returns $(C_{A_1})^{s_{A_2}}$, $(C_{B_1})^{s_{B_2}}$ to S_1. Based on the hardness of computing discrete logarithm, it is infeasible for S_1 to deduce s_{A_2} or s_{B_2}. Therefore, S_1 cannot recover s_A or s_B.

For the case of collusion between S_1 and MU_1, $\{\hat{x}_{1i}\}_{i=1}^{n_a}$ and $\{\hat{x}_{2i}\}_{i=1}^{n_a}$ are both revealed to $\mathcal{A}^{\mathbb{SH}}_{(S_1,MU_1)}$. Even though the adversary can know $\sum_{j=n_a+1}^{n} \hat{x}_{1j}\hat{x}_{2j}$, the value of $\{\hat{x}_{1j}\}_{j=n_a+1}^{n}$ and $\{\hat{x}_{2j}\}_{j=n_a+1}^{n}$ remains unknown. According to Theorem 3, it is secure to reveal gram matrix to S_1.

Step 3 Train SVM: Given the gram matrix K, we train SVM to get α. $\mathcal{A}^{\mathbb{SH}}_{S_1}$ cannot infer anything about gene expression profiles based on α.

Step 4 Reconstruct β based on α: If gene expression profiling microarray is known to the public, then $\mathcal{A}^{\mathbb{SH}}_{S_1}$ will know which genes are picked out as biomarkers corresponding to non-zero elements of β. We can remedy this vulnerability through permuting the gene expression profile before uploading.

References

1. Duffy, M.J., Crown, J.: A personalized approach to cancer treatment: how biomarkers can help. Clin. Chem. **54**(11), 1770–1779 (2008)
2. Barretina, J., Caponigro, G., Stransky, N., Venkatesan, K., Margolin, A.A., Kim, S., Wilson, C.J., Lehár, J., Kryukov, G.V., Sonkin, D., et al.: The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. Nature **483**(7391), 603–607 (2012)
3. Garnett, M.J., Edelman, E.J., Heidorn, S.J., Greenman, C.D., Dastur, A., Lau, K.W., Patricia Greninger, I., Thompson, R., Luo, X., Soares, J., et al.: Systematic identification of genomic markers of drug sensitivity in cancer cells. Nature **483**(7391), 570–575 (2012)
4. Covell, D.G.: Data mining approaches for genomic biomarker development: applications using drug screening data from the cancer genome project and the cancer cell line encyclopedia. PloS One **10**(7), e0127433 (2015)
5. Jang, I.S., Neto, E.C., Guinney, J., Friend, S.H., Margolin, A.A.: Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data. In: Pacific Symposium on Biocomputing, p. 63. NIH Public Access (2014)
6. Erlich, Y., Narayanan, A.: Routes for breaching and protecting genetic privacy. Nat. Rev. Genet. **15**(6), 409–421 (2014)
7. Schadt, E.E., Woo, S., Hao, K.: Bayesian method to predict individual SNP genotypes from gene expression data. Nat. Rev. Genet. **44**(5), 603–608 (2012)
8. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Roy. Stat. Soc. Ser. B (Methodol.) **58**, 267–288 (1996)
9. Friedman, J., Hastie, T., Tibshirani, R.: glmnet: lasso and elastic-net regularized generalized linear models. R Package Version, 1 (2009)
10. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. Roy. Stat. Soc. Ser. B (Stat. Methodol.) **67**(2), 301–320 (2005)
11. Liu, J., Ji, S., Ye, J., et al.: SLEP: sparse learning with efficient projections, vol. 6, p. 491. Arizona State University (2009)

12. Shalev-Shwartz, S., Zhang, T.: Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In: ICML, pp. 64–72 (2014)
13. Zhou, Q., Chen, W., Song, S., Gardner, J.R., Weinberger, K.Q., Chen, Y.: A reduction of the elastic net to support vector machines with an application to GPU computing. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
14. Lin, K.P., Chen, M.S.: Privacy-preserving outsourcing support vector machines with random transformation. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 363–372. ACM (2010)
15. Laur, S., Lipmaa, H., Mielikäinen, T.: Cryptographically private support vector machines. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 618–624. ACM (2006)
16. Tassa, T., Jarrous, A., Ben-Ya'akov, Y.: Oblivious evaluation of multivariate polynomials. J. Math. Crypt. **7**(1), 1–29 (2013)
17. Liu, F., Ng, W.K., Zhang, W.: Encrypted SVM for outsourced data mining. In: 2015 IEEE 8th International Conference on Cloud Computing, pp. 1085–1092. IEEE (2015)
18. Yu, H., Jiang, X., Vaidya, J.: Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data. In: Proceedings of the 2006 ACM Symposium on Applied Computing, pp. 603–610. ACM (2006)
19. Yu, H., Vaidya, J., Jiang, X.: Privacy-preserving SVM classification on vertically partitioned data. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS, vol. 3918, pp. 647–656. Springer, Heidelberg (2006). doi:10.1007/11731139_74
20. Vaidya, J., Hwanjo, Y., Jiang, X.: Privacy-preserving svm classification. Knowl. Inf. Syst. **14**(2), 161–178 (2008)
21. Van Dijk, M., Juels, A.: On the impossibility of cryptography alone for privacy-preserving cloud computing. HotSec **10**, 1–8 (2010)
22. Chow, S.S., Lee, J.H., Subramanian, L.: Two-party computation model for privacy-preserving queries over distributed databases. In: NDSS (2009)
23. Erkin, Z., Veugen, T., Toft, T., Lagendijk, R.L.: Generating private recommendations efficiently using homomorphic encryption and data packing. IEEE Trans. Inf. Forensics Secur. **7**(3), 1053–1066 (2012)
24. Demmler, D., Schneider, T., Zohner, M.: ABY-A framework for efficient mixed-protocol secure two-party computation. In: NDSS (2015)
25. Pedersen, T.B., Saygın, Y., Savaş, E.: Secret charing vs. encryption-based techniques for privacy preserving data mining (2007)
26. López-Alt, A., Tromer, E., Vaikuntanathan, V.: 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, pp. 1219–1234. ACM (2012)
27. Peter, A., Tews, E., Katzenbeisser, S.: Efficiently outsourcing multiparty computation under multiple keys. IEEE Trans. Inf. Forensics Secur. **8**(12), 2046–2058 (2013)
28. Wang, B., Li, M., Chow, S.S., Li, H.: Computing encrypted cloud data efficiently under multiple keys. In: 2013 IEEE Conference on Communications and Network Security (CNS), pp. 504–513. IEEE (2013)
29. Wang, B., Li, M., Chow, S.S., Li, H.: A tale of two clouds: computing on data encrypted under multiple keys. In: 2014 IEEE Conference on Communications and Network Security (CNS), pp. 337–345. IEEE (2014)

30. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998). doi:10.1007/BFb0054122
31. Catalano, D., Fiore, D.: Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1518–1529. ACM (2015)
32. Bresson, E., Catalano, D., Pointcheval, D.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003). doi:10.1007/978-3-540-40061-5_3
33. Yang, W., Soares, J., Greninger, P., Edelman, E.J., Lightfoot, H., Forbes, S., Bindal, N., Beare, D., Smith, J.A., Richard Thompson, I., et al.: Genomics of drug sensitivity in cancer (GDSC): a resource for therapeutic biomarker discovery in cancer cells. Nucleic Acids Res. **41**(D1), D955–D961 (2013)
34. Jaggi, M.: An equivalence between the Lasso and support vector machines. In: Argyriou, A., Signoretto, M., Suykens, J.A.K. (eds.) Regularization, Optimization, Kernels, and Support Vector Machines, pp. 1–26. Taylor & Francis, Boca Raton (2014)
35. Zhou, H., Wornell, G.: Efficient homomorphic encryption on integer vectors and its applications. In: Information Theory and Applications Workshop (ITA 2014), pp. 1–9. IEEE (2014)
36. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30576-7_18
37. Fushiki, T.: Estimation of prediction error by using k-fold cross-validation. Stat. Comput. **21**(2), 137–146 (2011)
38. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
39. Wong, W.K., Cheung, D.W.L., Kao, B., Mamoulis, N.: Secure kNN computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pp. 139–152. ACM (2009)
40. Halevi, S., Shoup, V.: An implementation of homomorphic encryption. https://github.com/shaih/HElib
41. Keerthi, S.S., Chapelle, O., DeCoste, D.: Building support vector machines with reduced classifier complexity. J. Mach. Learn. Res. **7**, 1493–1515 (2006)
42. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). doi:10.1007/3-540-46766-1_9
43. Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the 25th International Conference on Machine Learning, pp. 408–415. ACM (2008)
44. Tyree, S., Gardner, J.R., Weinberger, K.Q., Agrawal, K., Tran, J.: Parallel support vector machines in practice. arXiv preprint arXiv:1404.1066 (2014)
45. Chapelle, O.: Training a support vector machine in the primal. Neural Comput. **19**(5), 1155–1178 (2007)