Linear Model with Local Differential Privacy

Guanhong Miao, A. Adam Ding, and Samuel S. Wu

Abstract—Scientific collaborations benefit from collaborative learning of distributed sources, but remain difficult to achieve when data are sensitive. In recent years, privacy preserving techniques have been widely studied to analyze distributed data across different agencies while protecting sensitive information. Secure multiparty computation has been widely studied for privacy protection with high privacy level but intense computation cost. There are also other security techniques sacrificing partial data utility to reduce disclosure risk. A major challenge is to balance data utility and disclosure risk while maintaining high computation efficiency. In this paper, matrix masking technique is applied to encrypt data such that the secure schemes are against malicious adversaries while achieving local differential privacy. The proposed schemes are designed for linear models and can be implemented for both vertical and horizontal partitioning scenarios. Moreover, cross validation is studied to prevent overfitting and select optimal parameters without additional communication cost. Simulation results present the efficiency of proposed schemes to analyze dataset with millions of records and high-dimensional data ($n << p$).

Index Terms—Malicious adversary, local differential privacy, linear model, cross validation, vertical and horizontal partitioning, matrix masking.

Disclaimer This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

I. INTRODUCTION

The demand of collaborative learning over distributed datasets increases as recent advances in computing and communication technologies. Agencies cooperate to build statistical models on aggregated datasets to obtain more accurate models. Vertical and horizontal partitioning are two common partitioning approaches to integrate distributed datasets. Vertical partitioning happens when participating agencies have datasets with different sets of attributes on the same sets of subjects. For example, biomedical applications often need to consult records distributed among several heterogeneous domains, such as genotype data, clinical data and medical imaging, to define more accurate diagnosis for a single patient. Horizontal partitioning happens when multiple agencies have datasets with identical attributes for disjoint sets of subjects. In many cases data are collected over different sites with the same attributes. For instance, hospitals in different locations have the same type of diagnosis records and other health related information for different patients.

Privacy protection is a big challenge to perform collaborative learning as data may contain sensitive information so that data owners may not be willing to share data unless privacy is guaranteed. For instance, biomedical data integration and sharing raise public concerns that information exchange (e.g., demographics, genome sequences, medications) can put sensitive patient information at risk. A breach can have serious implications for research participants.

A variety of literatures have addressed diverse solutions for privacy preserving collaborative learning. Vaidya and Clifton [1] developed secure protocols to find association rules over the vertically partitioned data. Nikolaenko et al. [2] proposed a secure linear regression approach for a scenario where many parties upload their data to a server to build the model. A privacy preserving linear regression protocol was investigated for vertical partitioning on high-dimensional data [3]. Secure systems that work for both vertical and horizontal partitioning were presented in [4], [5]. Maliciously secure cooperative learning for horizontally partitioned linear models were proposed in [6].

In this paper, we develop privacy preserving schemes for linear models. Because linear models are easy to interpret and statistically robust, they are widely used in bioinformatics research [7], financial risk analysis [8], and are the foundation of basis pursuit techniques in signal processing. We investigate linear model schemes to achieve security against malicious adversaries (which means adversary may use any efficient attack strategy and thus may arbitrarily deviate from the protocol specification) with efficiency to permit use on relatively large datasets. Our contributions are as follows:

1) Our schemes are against malicious adversaries using zero-knowledge proofs. The malicious behavior of any party deviating from the proposed schemes can be detected. If any agency deviates from the schemes, the results are not accurate but still no sensitive information of original data is released.

2) The proposed schemes satisfy local differential privacy, such that the probability distribution of scheme output is roughly the same for any two inputs. The output does not reveal significant information about any particular element in the input.

3) The encryption schemes are proposed for both vertical and horizontal partitioning scenarios.

4) Cross validation is feasible in the proposed schemes to prevent overfitting problem and select penalty parameters in ridge regression without additional communication cost.

5) The schemes are efficient to analyze large datasets with millions of records and high-dimensional data ($n << p$).

The rest of the paper is organized as follows. Section
II. RELATED WORK

Privacy preserving data analysis falls into two major categories: perturbation-based approaches and secure multiparty computation (SMC)-based approaches. Differential privacy [9] has been widely embraced by research communities as an accepted notion of privacy for statistical analysis.

Data perturbation techniques have been widely studied as a tool of privacy preserving data mining [10]–[12]. Chen et al. developed geometric perturbation [11] and added noise term to enhance the security. Data utility is preserved using the geometric perturbation. The noise term drops the utility and is not ideal to build accurate models. Liu et al. proposed random projection perturbation [12] by dimension reduction approach. The dimension reduction approach loses some information of the data and large sample size is required in order to reach acceptable power. Moreover, plenty of studies focused on linear models using perturbation approaches to encrypt data. Linear regression based on matrix masking techniques were investigated for different privacy preserving problems [13]–[16]. Du et al. [13] studied linear regression in Secure 2-party Computation framework where each of the two parties holds a secret data set and wants to conduct analysis on the joint data. Karr et al. [14] used secure matrix product technique to allow multiple parties to estimate linear regression coefficients but was not immune to breaches of privacy. Wu et al. [15] investigated schemes to collect data privately granting data users access to non-sensitive personal information while sensitive information remains hidden. Matrix masking were investigated for privacy preserving techniques in [16]–[21]. [17] investigated secure outsourcing face recognition based on elementary matrix transformation. [18] studied secure algorithms for outsourcing linear equations. Secure outsourcing algorithms of matrix operations were proposed in [19]. In [21], matrix filled with random integers were used for encryption by both-sided matrix multiplication which ensures robustness to known plaintext attack and brute-force attack. Sparse matrix masking was used to design privacy preserving outsourced computation in [20]. Chen et al. [16] investigated efficient linear regression outsourcing to a cloud. The secure schemes were questioned for the vulnerability to disclosure attack and its research significance [22]. Due to the trade-off between data utility and disclosure risk, matrix masking methods proposed in previous studies face potential disclosure risks and may release extra information of original data under certain circumstances.

Plenty of previous works utilized cryptographic techniques and SMC to control disclosure risk [2]–[6], [23], [24] for secure linear models. By allowing the evaluation of arbitrary computations on encrypted data without decrypting it, homomorphic encryption (HE) schemes were predominantly applied in state-of-the-art SMC-based approaches. Hall et al. [23] proposed an iteration algorithm to compute the inversion of matrix privately for secure linear regression. Cock et al. [24] further improved the inversion protocol for the parties to compute linear regression coefficients cooperatively. Nikolaenko et al. [2] proposed a hybrid approach using garbled circuit method for a large distributed dataset among million of users. The major bottlenecks of this protocol are that the number of gates in the garbled circuit is large and the computation cost grows proportionally. Gascón et al. [3] extended protocol in [2] for vertically partitioned data distributed among agencies. Conjugate gradient descent was applied to provide a more efficient computation while maintaining accuracy and convergence rate. Maliciously secure linear model, Helen, was investigated for horizontal partitioning in [6]. Helen was designed for the cases that organizations have large amount of records (up to millions) and a smaller number of features (up to hundreds). Using homomorphic encryption and SMC protocols, Helen is able to achieve high level of privacy protection but also requires expensive computation cost.

Table I summarizes main references studying privacy preserving linear models. “K-party: Yes” refers to $K(>2)$ agencies can perform the computation with equal trust (do not need to include the two non-colluding servers model).

| Privacy scheme | K-party? | Maliciously secure? |
|----------------|---------|-------------------|
| [13]           | Yes     | No                |
| [25]           | Yes     | No                |
| [26]           | Yes     | No                |
| [14]           | No      | No                |
| [23]           | Yes     | No                |
| [2]            | No      | No                |
| [24]           | No      | No                |
| [3]            | No      | No                |
| [4]            | No      | No                |
| [5]            | No      | No                |
| [6]            | Yes     | Yes               |

Table I summarizes main references studying privacy preserving linear model in recent years. Apart from the two common secure categories above, a distributed computation algorithm for linear regression was given in [25]. The limitations of this method were also introduced such as the possible disclosure risk from the coefficients. Prior secure schemes did not provide malicious security except [6] and the training process in most of them require outsourcing to two non-colluding servers. Privacy preserving ridge regression has also been investigated previously [2], [3], [6] focusing on $n > p$ problem. Notably, computation burdens are bottlenecks of previous secure linear models. More specifically, approaches based on HE cryptosystems involve an encoding mechanism, i.e., scaling, that converts floating-point numbers with fixed precision to integers. Larger scaling factors yield larger encryption parameters and worse performance while smaller scaling factors yield smaller encryption parameters and better performance but outputs may vary beyond the tolerance...
and lead to prediction inaccuracy [27]. Moreover, SMC-based approaches expect the data owners to be online and participate in the computation throughout the entire process.

Privacy preserving linear model achieving differential privacy has also been investigated [28]–[32]. [28] enforced differential privacy by perturbing the objective function of the optimization problem. [30] reduced the dimension of attributes to achieve differential privacy while [31] reduced the dimension of subjects for differential privacy. [32] built an local differential privacy-compliant stochastic gradient descent algorithm. All these studies focused on analyzing single dataset instead of collaborative learning. Moreover, dimension reduction (either dimension of subjects or attributes) disables cross validation or deriving model estimates for each attribute.

In this paper, we propose secure and efficient linear models for collaborative learning enabling practical implementation for high-dimensional data analysis. The proposed schemes are against malicious adversary while satisfying differential privacy.

III. PRELIMINARIES

A. Linear model

The linear regression model is

\[ Y = X\beta + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I) \]

where \( Y \in \mathbb{R}^n \) is a vector of responses, \( X \in \mathbb{R}^{n \times p} \) is the design matrix (categorical variables are transformed to dummy variables), \( \beta \) is a \( p \times 1 \) vector of regression coefficients including the intercept, \( \epsilon \) is an \( n \times 1 \) vector of random errors and \( \mathcal{N} \) denotes multivariate normal distribution. The estimates for \( \beta \) is \( \hat{\beta} = (X^T X)^{-1} X^T Y \).

Ridge regression, linear regression with \( L_2 \)-regularized penalty, is usually used to do variable selection for high-dimensional datasets [33]. It minimizes the residual sum of squares subject to a bound on the \( L_2 \)-norm of the coefficients

\[ \hat{\beta}_{\text{ridge}} = \arg\min_{\beta} \{ (Y - X\beta)^T (Y - X\beta) + \lambda \beta^T \beta \}. \]

Ridge solutions are given by \( \hat{\beta}_{\text{ridge}} = (X^T X + \lambda I)^{-1} X^T Y \) where \( \lambda \) is a tuning parameter.

B. Singular value decomposition (SVD)

The SVD of \( n \times p \) matrix \( X \) has the form

\[ X = UDV^T \]

where \( U \) and \( V \) are \( n \times p \) and \( p \times p \) orthogonal matrices. \( D \) is a \( p \times p \) diagonal matrix with diagonal entries \( d_1 \geq d_2 \geq \cdots \geq d_p \geq 0 \) called the singular values of \( X \). \( X^T X = V D^2 V^T \) is the eigen decomposition of \( X^T X \). The eigenvectors \( v_j \) (columns of \( V \)) are also called the principal components directions of \( X \). \( z_j = X v_j \) is the \( j \)-th principal component of \( X \) and \( P_C = XV \) contains the principal components of \( X \).

C. Local differential privacy

A randomized function \( f \) satisfies \( \epsilon \)-local differential privacy if and only if for any two inputs \( t \) and \( t' \) in the domain of \( f \), and any \( s \subseteq S \) where \( S \) contains \( f \)'s all possible output,

\[ P(f(t) \in s) \leq e^\epsilon P(f(t') \in s). \]

D. Zero-knowledge proofs

Zero-knowledge proofs convert a semi-honest protocol to a malicious-secure protocol by incorporating a proof that the protocol is executed correctly [34]. It is defined as proofs that convey no additional knowledge other than the correctness of the proposition in question. Let \( \pi \) be the semi-honest protocol. Zero-knowledge proofs ensure that a malicious party can either run \( \pi \) honestly, or cheat in \( \pi \) but cause the zero-knowledge proof to fail.

IV. SYSTEM OVERVIEW

A. System model

This paper focuses on collaborative learning in which data is stored by different agencies locally and they try to build linear regression models using all the data while preserving data confidentiality. Suppose there are \( K \) agencies and \( X_i \) is held by agency \( i (i = 1, \ldots, K) \). For vertical partitioning scenario, agencies has the same set of subjects and different sets of attributes and response \( Y \) is held by one agency. For horizontal partitioning scenario, agencies has the same set of attributes and different sets of subjects and agency \( i \) holds its own response \( Y_i \).

As shown in Figure 1, the proposed privacy preserving scheme can be separated into pre-modeling, modeling and post-modeling phase. In the pre-modeling phase, data are encrypted by the corporation among agencies. Encrypted data are then sent to cloud computing service provider (i.e., cloud server). The cloud server can be chosen among agencies. In the modeling phase, the cloud server conducts privacy preserving linear models using encrypted data. The cloud server then sends encrypted model results back to agencies. In the post-modeling phase, agencies cooperate to decrypt model results.

B. Design goals

The design goals are summarized as follows.

1. Correctness: The user can obtain the correct answer if agencies and the cloud server follow the scheme properly.

2. Privacy: The cloud server cannot derive any sensitive information from encrypted data nor from the collusion with agencies.

3. Soundness: Agencies are able to verify whether the returned result is correct.

4. Efficiency: The secure scheme is efficient to analyze data with millions of records or attributes.
C. Framework

We implement collaborative learning following the well-established privacy preserving outsourced computing framework. Given a computation request $F$, the general framework of a secure outsourcing computation scheme includes four sub-algorithms.

1. **KeyGen($F$):** With the input of the computation request $F$, agencies run the algorithm to generate a secret key SK for subsequent encryption and decryption.

2. **ProbGen(SK, $M$):** Given the corresponding input $M$, agency uses the secret key SK to encrypt $M$ into ciphertext matrix $M^*$ and sends $M^*$ to other agencies.

3. **ProbSolve($M^*$):** Receiving the encrypted $M^*$ from agencies, the cloud computes model estimates $\hat{\beta}^*$ and sends to agencies.

4. **ResultDec($\hat{\beta}^*$):** Upon input of the result $\hat{\beta}^*$, agencies decrypt it into $\hat{\beta}$ utilizing the secret key SK. An essential step is then applied to verify if the cloud server or agencies follow the scheme.

The above four steps can be summarized into three phases in our proposed scheme. *KeyGen($F$) and ProbGen(SK, $M$) are performed in the pre-modeling phase, ProbSolve($M^*$) is conducted in the modeling phase and ResultDec($\hat{\beta}^*$) is done in the post-modeling phase.*

D. Threat model

Assume that adversary model is malicious, i.e., agencies or the cloud server may arbitrarily deviate from the protocol specification and use any efficient attack strategy. They may intentionally return a random or forged result. We consider a strong threat model in which all but one agency/cloud server can be compromised by a malicious attacker. More specifically, the compromised agencies/cloud server can deviate arbitrarily from the proposed scheme, such as executing different computation than expected.

**Out of scope attacks:** The proposed schemes do not prevent a malicious agency from inputing a bad dataset for the computation in attempt to alter model result (i.e., poisoning attack [35]). However our zero-knowledge proofs ensure that once an agency provides an input into the computation, the agency is bound to using the same input consistently throughout the entire computation.

V. THE PROPOSED SCHEME

A. Pre-modeling phase

Without loss of generality, $(1, \cdots, 1)^T$ is added to $X$ which functions as intercept. Assume the intercept is held by agency 1 for vertical partitioning. For horizontal partitioning, every agency adds a vector of 1’s as one attribute in its dataset. Attributes are normalized before data transfer among agencies. Normalization is conducted locally by each agency for vertical partitioning. For horizontal partitioning, secure summation protocol [36] is applied to get overall mean and variance for each attribute.

Two encryption layers are designed to mask data. Additive noise $\Delta$ is added to original dataset to get first layer encrypted data $X^* = X + \Delta$. Noise addition has higher data utility compared with other privacy preserving approaches [37]. However, additive noise is vulnerable to disclosure attacks [38]-[41]. In the second encryption layer, data is further encrypted by row and column transformation: $X^* = AX*B$ and $Y^* = AY$ ($A$ and $B$ are randomly generated orthogonal and invertible matrix, respectively). To summarize, the mechanism $g$ is $g(X) = A(X + \Delta)B$. Furthermore, two pseudo responses are generated with one for verification and another to enhance encryption. We define the first pseudo response $Y_{s1} = \sum_{i=1}^{p} x_i$ where $x_i$ is the $i$-th column in $A(X + \Delta)B$. The second response $Y_{s2}$ is generated randomly. Let $Z = [Y, Y_{s1}, Y_{s2}]C$ where $C$ is a $3 \times 3$ invertible matrix and the final encrypted response matrix would be $Z^* = AZ$.

We design separate encryption schemes for vertical and horizontal partitioning in order to guarantee high level privacy preservation and maintain high data utility. Any matrix is masked by the data owner before sending out for analysis. Suppose agency $i$ holds $X_i$ and $Y_i$. In general, agency $i$ generates $A_i, B_i, C_i$, and sends out $A_i(X_i + \Delta_i)B_i, A_i[Y_i, Y_{s1}, Y_{s2}]C_i$ as outsourcing data. Then agency $j$ generates $A_j, B_j$ and $C_j$.
to mask received data. Matrices are masked by all agencies using similar methods.

1) First layer encryption
Given \( n \times p \) original data \( X_i \) \((i = 1, \ldots, K)\), generate noise matrix \( \Delta_i \) with each element \( e \) independent of \( X_i \) following Gaussian distribution \( N(0, \sigma^2) \). \( \Delta \) is added to \( X_i \) to get encrypted data \( X_i^* = X_i + \Delta_i \).

2) Second layer encryption
a) Vertical partitioning
In the second layer encryption, \( X_i^* \) \((i = 1, 2, \ldots, K)\) is further encrypted by agency \( i \) by row and column transformation. Since agencies use different masking matrices, we need to further regulate the encryption approach for data utility. In order to maintain data utility for linear regression models, we need to encrypt each agency \( i \) encrypts its data, \( X_i^* \), row and column transformed with each element \( e \) independent of \( X_i \) following Gaussian distribution \( N(0, \sigma^2) \), and sends to agency \( i \) to get the encrypted data \( X_i^* = X_i + \Delta_i \).

Agencies cooperate to finish the pre-modeling phase. To make \( A_i \) commutative, each agency first generates \( A_0 \) locally as the matrix basis using the same random seed. Then agency \( i \) generates \( A_i = A_0^{\gamma_i} \) where \( \gamma_i \) is a positive integer randomly generated \((i = 1, 2, \ldots, K)\). Suppose agency \( i \) has \( n \) subjects and \( p_i \) attributes \((i = 1, 2, \ldots, K)\). Agency \( i \) generates orthogonal matrix \( A_i \) and invertible matrix \( B_{i1}, B_{i2}, \ldots, B_{iK} \) \((i = 1, 2, \ldots, K)\). The dimension of \( A_i \) is \( n \times n \) and the dimension of \( B_{i1}, B_{i2}, \ldots, B_{iK} \) is \( p_1 \times p_1, p_2 \times p_2, \ldots, p_K \times p_K \), respectively \((i = 1, 2, \ldots, K)\). Let \( X_i^* = A_i(X_i + \Delta_i)B_{ii} \) and agency \( i \) releases \( X_i^* \) to other agencies. Agency \( j \) \((j \neq i)\) then masks it with \( A_j \) and \( B_{ji} \) and releases \( A_jX_i^*B_{ji} \) to other agencies. Finally, \( X_i^* \) are masked by all agencies in a pre-specific order. Suppose response \( Y \) is only known by agency 1. Each agency sums up the variables held by itself and then sends to agency 1 to get the first pseudo response \( Y_{s1} \). Then agency 1 randomly generates \( Y_{s2} \) and \( 3 \times 3 \) invertible matrix \( C \) to get \( Z = [Y_{s1}, Y_{s2}]C \). \( Z \) is masked by each agency using the left orthogonal masking matrix.

Table II summarizes masking matrices each agency needs to generate for data encryption. Algorithm 1 gives details of pre-modeling procedures for \( K \) agencies assuming agency 1 holds response \( Y \). Figure 2 presents an example of the masking procedures with a specific masking order.

If the communication order among agencies is
In order to maintain data utility for linear models, we need to further regulate the encryption approach for data. Different masking matrices are used by different agencies and the masking matrix generated by agency \( i \) is commutative. Each agency does not know masking matrix generated by other agencies, so the release of data is done in a pre-specific order.

The computation complexity of matrix generation and multiplication increases when the dimension of dataset increases. It is computational expensive when \( n \) is large. One way to reduce the cost is to partition \( A_i \) into block diagonal matrix. It is the same to partition \( A_0 \) since \( A_i \) is generated using matrix basis \( A_0 \). For example, there are 10,000 subjects in a dataset. If \( A_0 \) is partitioned with block size 100, each agency generates 100 orthogonal matrices with dimension 100 \( \times \) 100 instead of one 10,000 \( \times \) 10,000 matrix. In other words, \( A_i = \text{diag}(A_{i1}, \ldots, A_{i100}) \) where \( A_i \) is a random orthogonal matrix. The same strategy can be applied when \( p \) is big and agencies use partitioned block diagonal matrix for masking.

b) Horizontal partitioning

The first layer encrypted data \( X_i^* \) is further encrypted by agency \( i \) from left side and right side. Different masking matrices are used by different agencies and we need to further regulate the encryption approach for data utility. In order to maintain data utility for linear models, \( X_i^* \) needs to be encrypted by identical \( B \) after encryption procedure since the aggregated dataset is in the form of \( [X_1^T, X_2^T, \ldots, X_K^T] \) and \( [Y_1^T, Y_2^T, \ldots, Y_K^T] \).

Because each agency does not know masking matrix generated by other agencies, \( X_i^* \) should be encrypted by all agencies and \( B_i \) (the right masking matrix generated by agency \( i \)) is requested to be commutative. This specific encryption approach guarantees that \( X_i^* \) is encrypted by identical \( B \).

In the pre-modeling phase, agencies cooperate with each other. To make \( B_i \) commutative, each agency first generates \( B_0 \) locally as the matrix basis using the same random seed. Then agency \( i \) generates a random positive integer \( s_i \) random coefficient vector \( (b_{i1}, \ldots, b_{is_i}) \) and computes \( B_i = \sum_{j=1}^{s_i} b_{ij} B_0 \) for \( (1, 2, \ldots, K) \). Suppose agency \( i \) has \( n_i \) subjects and \( p \) attributes \( (i = 1, 2, \ldots, K) \). Agency \( i \) generates orthogonal matrix \( A_{i1}, A_{i2}, \ldots, A_{iK} \), invertible matrix \( B_i \) \( (i = 1, 2, \ldots, K) \) and 3 \( \times \) 3 invertible commutative matrix \( C_i \) using matrix basis \( C_0 \). The dimension of \( B_i \) is \( p \times p \) and the dimension of \( A_{i1}, A_{i2}, \ldots, A_{iK} \) is \( n_i \times n_i, n_2 \times n_2, \ldots, n_K \times n_K \), respectively. The masking matrices generated for data encryption are listed in Table III. Detailed procedures and example of the masking are illustrated in Appendix A (Algorithm 3 and Figure 7).

If the communication order among agencies is \( 1 \rightarrow 2 \rightarrow \ldots \rightarrow K \rightarrow 1 \), the final released dataset would be \( A'X^*B = A_1 A_2 \cdots A_K [X_1^*, X_2^*, \ldots, X_K^*]B \) where \( B = \begin{pmatrix} B_{11} B_{21} \cdots B_{K1} \\ B_{22} B_{22} B_{23} \cdots \cdots B_{KK} B_{K2} B_{K3} \cdots B_{(K-1)K} \end{pmatrix} \) as \( A_i \) is commutative. Final released response matrix would be \( AZ = A_1 A_2 \cdots A_K Z \).

### TABLE II: Masking matrices for vertical partitioning. Assume agency 1 holds response \( Y \). Each agency generates matrices listed in the corresponding row. Each dataset \( X_i^* \) (\( i = 1, 2, \ldots, K \)) is masked by matrices listed in the corresponding column.

| Masking role | Agency 1 | Agency 2 | Agency K |
|--------------|----------|----------|----------|
| (A1, B1)    | (A1, B12) | (A1, B1K) | (A1, B1K) |
| (A2, B21)   | (A2, B22) | (A2, B2K) | (A2, B2K) |
| (A3, B31)   | (A3, B32) | (A3, B3K) | (A3, B3K) |

### TABLE III: Masking matrices for horizontal partitioning. Each agency generates matrices listed in the corresponding row. Each dataset \( X_i^* \) and \( Y_i \) are masked by matrices listed in the corresponding column.

| Masking role | Agency 1 | Agency 2 | Agency K |
|--------------|----------|----------|----------|
| (A11, B11, C1) | (A12, B12) | (A12, B1K) | (A1K, B1K) |
| (A21, B21)   | (A22, B22, C2) | (A22, B2K) | (A2K, B2K) |
| (A31, B31)   | (A32, B32) | (A32, B3K) | (A3K, B3K) |

The masking matrices generated for data encryption are listed in Table III. Detailed procedures and example of the masking are illustrated in Appendix A (Algorithm 3 and Figure 7).

If the communication order among agencies is \( 1 \rightarrow 2 \rightarrow \ldots \rightarrow K \rightarrow 1 \), the final released dataset would be \( A'X^*B \) and \( AZ \) where \( Z = [Z_1^T, Z_2^T, \ldots, Z_K^T] \). By following the proposed scheme. Let \( \beta_1, \beta_2, \ldots, \beta_K \) be the estimates for attributes of agency \( i \) and \( \beta_1 \) is a vector of 1’s if each agency follows the proposed scheme. Let \( \beta = (\beta_1, \beta_2, \ldots, \beta_K) \) and \( B = B_1 B_2 \cdots B_K \) as \( B_i \) is commutative.

For high dimensional data, apply the same strategy discussed in Section V-A2a to improve computation efficiency.

### B. Modeling and post-modeling phase

1) \( \Delta = 0 \)

We first analyze model results for special case \( \Delta = 0 \). Using encrypted dataset \( X^* = AXB \) and \( Z^* = A[Y, Y_{s1}, Y_{s2}]C \) under vertical partitioning scenario, we get \( \beta^* = (X^T X)^{-1} X^T Z^* = B^{-1}(X^T X)^{-1} X^T Z \) and \( \beta^* = B^{-1} \beta C = B^{-1}[\beta_1, \beta_2, \ldots, \beta_K] \) where \( \beta \) contains the estimates for the true response and \( \beta_1 \) should be a vector of 1’s if each agency follows the proposed scheme. Let \( \beta = (\beta_1, \beta_2^*, \ldots, \beta_K^*) \) with \( \beta_i \) be the estimates for attributes of agency \( i \). We use the example given in Section V-A2a to show how to decrypt model estimates. Since \( \beta_1 = B_11 B_21 \cdots B_{K1} \beta_1 C^{-1} \) is sent to agencies with the order of \( K \rightarrow \cdots \rightarrow 2 \rightarrow 1 \) to eliminate \( B_{11} B_{21} \cdots B_{K1} \) (shown in Figure 3). \( C^{-1} \) is removed by agency who holds the true response. Agency 1 can choose not to publish \( \beta_1 \) since the final multiplied matrix \( B_{11} \) is held by itself. Same approach applies to get \( \beta_i \) (\( i = 2, \ldots, K \)).
where $Y_i$ contains Agency $i$’s three responses (true response and two pseudo responses). $\hat{\beta}^* = (X^*X^*)^{-1}X^*Z^* = B^{-1}[\hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_K] \left( \begin{array}{ccc}
C_1 \\
C_2 \\
\vdots \\
C_K \end{array} \right)$ where $\hat{\beta}_i$ is a $p \times 3$ matrix and the true model estimate $\beta = \sum_i^{K} \hat{\beta}_i$. To compute $\hat{\beta}$, $\hat{\beta}^*$ is sent to each agency in order to eliminate $B$ and $C_i$. The order of agencies is not important since $B_i$ ($i = 1, 2, \cdots, K$) is commutative. Detailed procedure of $\hat{\beta}_1$ decryption is given in Figure 3. Same approach applies to get $\hat{\beta}_i$ ($i = 2, \cdots, K$) with $C_i$ decrypted by Agency $i$. Then $\{\hat{\beta}_i; i = 1, \cdots, K\}$ is summed up to get $\hat{\beta}$.

Algorithm 4 (Appendix B) gives general post-modeling procedures for $K$ agencies. For vertical partitioning, agency $i$ can decide whether to publish $\hat{\beta}_i$ or not.

2) $\Delta \neq 0$

When $\Delta \neq 0$, $\hat{\beta}_\Delta^* = B^{-1}((X + \Delta)^T(X + \Delta))^{-1}(X + \Delta)^T[Y_i, Y_{s1}, Y_{s2}]C$. The decryption approach is the same as given above. After decryption, $\beta_\Delta = ((X + \Delta)^T(X + \Delta))^{-1}(X + \Delta)^T[Y_i, Y_{s1}, Y_{s2}]$. The association between $\sigma^2$ ($\sigma$ is the parameter of the normal distribution for additive noise matrix generation) and model accuracy is further investigated.

Privacy preserving ridge regression: In order to compute $\hat{\beta}_{ridge}^*$, matrix $B^TB$ needs to be computed and released additionally from pre-modeling phase using similar procedures in Algorithm 1 (vertical partitioning) or 3 (horizontal partitioning, Appendix A). For a given $\lambda$, we have $\hat{\beta}_{ridge}^* = [X^TX^* + \lambda(B^TB)^{-1}]^{-1}X^*Z^* = B^{-1}\hat{\beta}_{ridge}^*C$ from encrypted datasets. So $\hat{\beta}_{ridge} = B\hat{\beta}_{ridge}^*C^{-1}$. Apply Algorithm 4 (Appendix B) to get $\hat{\beta}_{ridge}$. Since different $\lambda$ yields different model estimates using previous defined pseudo response $Y_{s1} = \sum_{i=1}^{P} x_i$, let $Y_{s1}$ be a vector of $0$’s for ridge regression for verification to against malicious adversary.

Computation correctness: After decryption, $\hat{\beta} = [\hat{\beta}_s, \hat{\beta}_s, \hat{\beta}_{s2}]$ is a $p \times 3$ matrix for linear regression with $\hat{\beta}$ the true estimates for $Y$, $\hat{\beta}_s$ be the estimates for $Y_{s1}$ and $\hat{\beta}_{s2}$ be the estimates for $Y_{s2}$. Since $Y_{s1} = \sum_{i=1}^{p} x_i$, $\hat{\beta}_s = 1$ if each agency follows the proposed scheme. For ridge regression, $Y_{s1} = 0$ and $\hat{\beta}_s = 0$. To conclude, $\hat{\beta}_s$ is used to detect if any agency deviates from the schemes.

C. Cross validation

Cross validation is widely used to prevent overfitting problem. It is also the golden standard to select optimal $\lambda$ for ridge regression. By partitioning orthogonal matrix $A$, the proposed scheme enables cross validation for privacy preserving linear models without additional communication cost. The procedure of partitioning $A$ has been illustrated in Section V-A2a. We give examples for $k$-cross validation. For vertical partitioning, the matrix basis $A_0$ is partitioned into $k$ parts. In other words,

$$A_0 = \begin{pmatrix}
A_{01} & A_{02} & \cdots \\
& & \\
& & \\
& & \\
& & \\
& & \\
A_{0k} & \end{pmatrix}.$$ 

Subjects in each block are masked separately and can be used as training set or testing set. For horizontal partitioning, each agency uses $k$-blocked orthogonal matrix to mask data and subjects within each block are used as training set or testing set.

Our schemes are efficient and practical to change regularization parameter $\lambda$ comparing with previous secure ridge regression protocols with $\lambda$ public and fixed [2], [3], [6], [23].

D. Privacy preserving principal component analysis (PCA)

First consider the scenario $\Delta = 0$. For encrypted data $X^* = AXB$, we have $B^TX^TXB = B^TV^DV^TB$. Since congruent matrices $X^TX$ and $B^TX^TXB$ have same eigenvalues, $V^* = B^TV$. Denote the encrypted principal component (PC) matrix $P^*_C = X^*(B^TB)^{-1}V^* = AXB(B^TB)^{-1}B^TV = AXV$. To compute $P^*_C$, matrix $B^TB$ needs to be computed and released additionally from pre-modeling phase using similar procedures in Algorithm 1 (vertical partitioning) or 3 (horizontal partitioning, Appendix A). For $X^* = A(X + \Delta)B$, $P^*_C = A(X + \Delta)V^*_\Delta$ where orthogonal matrix $V^*_\Delta$ derived from $(X + \Delta)^T(X + \Delta)$ is $V^*_\Delta D^2 V^*_\Delta$.

To get the first $j$ PCs, the first $j$ columns of $P^*_C$ is sent to each agency to eliminate the masking matrix $A$. 

![Fig. 3: Post-modeling procedure (top: vertical partitioning assuming $C$ is generated by Agency 1, bottom: horizontal partitioning).](image-url)
Algorithm 2: Privacy preserving PCA

Input: $X^*$, $B^T B$
Output: Encrypted PC matrix $P_C^*$
1) Derive $V^*$ from $X'^T X^* = V^* D V^T$; 
2) Compute $P_C^* = X^* (B^T B)^{-1} V^*$;

VI. SECURITY ANALYSIS

Without loss of generality, we use $X$ to denote the original data that contain sensitive information, $A$ and $B$ to denote the left and right masking matrices, respectively.

Matrix encryption has been widely used in previous privacy preserving studies. Privacy methods with encrypted matrix in the form of $XB$ and $AX$ were studied in [11]–[13], [15]. $XB$ and $AX$ have high data utility but the disclosure risks are also high. The potential disclosure risk of $XB$ and $AX$ are summarized in Table IV. Furthermore, sparse matrix masking with encrypted matrix in the form of $AXB$ ($A$ and $B$ were sparse matrices) were investigated in [16]–[20]. The privacy guarantee needs to be argued carefully for the sparse matrices masking as discussed in [18], [19]. Matrix encrypted from both sides has been proofed to solve different problems without releasing sensitive information in previous works [17]–[21].

| Masked data  | Attack technique |
|--------------|------------------|
| $AX$         | Covariance matrix |
| $XB$ or $AX$ | Known plaintext attack |

Relating these results to our schemes, we use dense orthogonal/invertible matrix for encryption and the density of our masking matrix ensures privacy protection. The released data are all encrypted from both left side and right side.

A. Infinite support

We first prove that no data information is disclosed to the general attacker who does not know basis matrices $A_0$ and $B_0$ for commutative matrix generation. Then we consider the disclosure risk assuming $A_0$ and $B_0$ are known, e.g., some participating agencies try to recover sensitive information from received data or the attacker colludes with some agencies.

Definition 1. $OI$-equivalent: Matrix $X$ is $OI$-equivalent to $\tilde{X}$ if there exists an orthogonal matrix $U$ and an invertible matrix $V$ such that $\tilde{X} = UXV$.

Lemma 1. Vector $x$ is $OI$-equivalent to $(1,0,\ldots,0)^T$.

Proof. Let $x^T x = c^2$, $V = 1/c$ and $U = (u_1, u_2, \ldots, u_n)^T$ where $u_1 = x/c$, $u_i^T u_i = 1$ and $u_i$ is orthogonal to $\{u_1, \ldots, u_{i-1}\}$ ($i = 2, \ldots, n$). We have $UxV = (c, 0, \ldots, 0)^T$. Since $u_1^T u_1 = x^T x/c^2 = 1$ and $u_i$ is orthogonal to $u_j$ ($i \neq j$), $U$ is orthogonal. By left orthogonal transformation and right invertible transformation, $x$ is $OI$-equivalent to $(1, 0, \ldots, 0)^T$.

Theorem 1. Suppose $X$ is equivalent to $X_0 = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}$. The support of encrypted matrix $AXB$ contains matrices that are $OI$-equivalent to $X_0$.

Proof. By the right invertible matrix transformation, encrypted matrix $AXB$ is $OI$-equivalent to its reduced column echelon form $(H, 0)$ where $H$ is an $n \times r$ matrix. Based on Lemma 1, there exists an orthogonal matrix $U_1$ and a diagonal matrix $V_1 = \text{diag}(0, \ldots, v_1, \ldots, 0)$, s.t. $U_1 (H, 0) V_1 = (H_1, 0)$ where the last column in $H_1$ is $(0, \ldots, 0, 1)^T$. There exists an invertible $r \times r$ matrix $V_2$, s.t., $(H_1, 0) \begin{pmatrix} V_2 & 0 \\ 0 & 1 \end{pmatrix} = (H_2, I_1)$ where $I$ is identity matrix, $I_1$ contains 0 except the first column being $(0, \ldots, 0, 1)^T$. $H_2$ is a $n \times (r - 1)$ matrix with the last row all zeros. Similarly, there exists a $(n - 1) \times (n - 1)$ orthogonal matrix $U_2$ and a diagonal matrix $V_3 = \text{diag}(0, \ldots, v_3, \ldots, 0)$, s.t. $U_2 \begin{pmatrix} V_2 & 0 \\ 0 & 1 \end{pmatrix} (H_2, I_1) V_3 = (H_3, I_1)$ where the last column in $H_3$ is $(0, \ldots, 0, 1, 0)^T$. After repeating these similar transformation steps, $AXB$ transforms to $\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}$ using $U = \prod U_i$ and $V = \prod V_j$. So $AXB$ is $OI$-equivalent to $\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}$.

Remark 1. Given $AXB$, attacker with no information of masking matrix basis $A_0$ and $B_0$ (used for commutative matrix generation) is impossible to recover $X$.

Consider some agencies try to derive sensitive information from received datasets. We show that using the same matrix basis to generate commutative invertible masking matrices (horizontal partitioning) does not increase the disclosure risk. Suppose the matrix basis is $B_0$. $B_{11} = \sum_{j=1}^{n_0} b_{1j} B_0^j$ and $B_{22} = \sum_{j=1}^{n_0} b_{2j} B_0^j$ are commutative matrices held by agency 1 and 2, respectively.

As pointed out in Section 3.3 of [42], a polynomial $f(t)$ is said to annihilate matrix $B$ if $f(B) = 0$. The minimal polynomial of $B$ is the monic polynomial of minimum degree that annihilates $B$. The minimal polynomial $f_B(t)$ is unique and $f_B(\lambda) = 0$ if and only if $\lambda$ is an eigenvalue of $B$. For given matrix $B$ and any monic polynomial $f(t)$, $f(B) = 0$ if and only if there exists monic polynomial $h(t)$ such that $f(t) = h(t)f_B(t)$ where $f_B(t)$ is the minimal polynomial of $B$. For polynomial $f(t)$ such that $f(B) = 0$, all the eigenvalues of $B$ are the roots of $f(t)$.

Theorem 2. $B_{ii}$ is determined by $b_{ij}$ ($j = 1, \ldots, s_0$) where $B_{ii}$ is generated by procedures given in Section V-A2b.

Proof. Let $f_1(t) = \sum_{j=1}^{s_0} b_{1j} t^j$, $f_2(t) = \sum_{j=1}^{s_0} b_{2j} t^j$ and $f(t) = f_1(t) - f_2(t)$. We have $B_{11} = f_1(B_0)$ and
The proposed schemes protect against a strong threat model in which all but one party can be compromised by a malicious attacker. Since data is encrypted by each agency before sending to other agencies, the malicious attacker cannot be able to learn anything about the data held by agency who does not collude other than the final result.

**Poisoning attack:** The attacker can inject poisoned inputs into a dataset before training in order to derive sensitive information. One way to detect such attacks is to check whether the model estimates \( \gamma_i \) (\( i = 1, \cdots, K \)) within acceptable ranges. By checking if the model estimates are within certain ranges (e.g., 95% CIs) for different training information. One way to detect such attacks is to check whether the model estimates are within certain ranges (e.g., 95% CIs) for different training datasets in cross validation, the proposed schemes protect against strong outliers, maliciously or erroneously inputs.
which can significantly distort the results. The parties can still input incorrect values, but the influence on the final result is limited.

VII. PERFORMANCE EVALUATION

A. Efficiency

In the proposed schemes, data encryption and decryption in the pre-modeling and post-modeling phase contribute to the increasing cost while the modeling phase has the same cost as non-private model computation.

**TABLE V: Computation summary.**

| Phase             | Involved agency |
|-------------------|-----------------|
| Pre-modeling      | Each agency     |
| Modeling*         | Server          |
| Post-modeling     | Each agency     |

*The computation cost of modeling phase is the same as non-secure model cost.

We conduct simulations to show the efficiency under two scenarios: \( n > p \) and \( n < p \). Assume there are \( K \) agencies and final aggregated dataset has \( n \) subjects and \( p \) attributes. Each agency has \( n \) subjects and \( p/K \) attributes under vertical partitioning scenario while each agency has \( n/K \) subjects and \( p \) attributes under horizontal partitioning scenario. Masking matrix \( A \) is generated as block diagonal matrix with block size \( \hat{n} = 100 \) to control the computation cost when the number of subjects held by agencies is large. All the experiments are performed in Matlab on University of Florida Hipergator 3.0 with 1 CPU with RAM determined by the size of data matrix.

Figure 4 and 5 present the extra computation cost of privacy preserving schemes beyond non-secure models for \( n > p \) and \( n << p \) scenario, respectively.

In the modeling phase, computing the inverse in the closed form solution of linear regression can be computationally expensive for high-dimensional dataset. Simulations show that it takes 3 minutes to compute linear regression estimates using closed form and less than 1 second to get the estimates using conjugate gradient descent for a dataset with \( 10^6 \) subjects and \( 10^3 \) attributes. We use gradient descent to compute linear regression estimates for high dimensional data.

Integrating the computation costs from pre-modeling, modeling and post-modeling phase, our proposed scheme is computationally efficient to analyze high dimensional dataset with 3 minutes or less to implement secure linear regression for datasets with millions of subjects.

The communication complexity for privacy preserving linear regression is \( O(Knp + Kp) \), approximately \( K \) times of the aggregated dataset.

B. Tradeoff between data utility and privacy

We set the number of subjects \( n = 10^4 \) and the number of attributes \( p = 500 \). We first draw \( X \in \mathbb{R}^{n \times p} \) following \( N(0, I) \) where \( I \) is the identity matrix. Let \( \beta \) be a \( p \)-dimensional vector where the first 100 estimates are chosen randomly from \( \pm b \) with \( b \sim N(2,1) \) and the other elements are zeros. Finally, we draw \( y = X\beta \). To evaluate the noise effect for linear regression estimates, the encrypted data is generated as \( X^* = X + \Delta \) where \( \Delta \sim N(0, \sigma^2) \).

Five metrics are used to compare linear regression by \( X \) and \( X^* \). Pearson’s correlation coefficient \( \rho \), Manhattan

![Fig. 4: Computation cost for \( n > p \) scenario. \( n \) denotes the number of subjects, \( p \) denotes the number of attributes and \( K \) denotes the number of agencies. \( K = 2 \) in the first and third figure and \( p = 240 \) in the second and fourth figure.](image)

![Fig. 5: Computation cost for \( n < p \) scenario. \( n \) denotes the number of subjects, \( p \) denotes the number of attributes and \( K \) denotes the number of agencies. \( K = 10 \) in the first and third figure and \( p = 10^6 \) in the second and fourth figure.](image)
distance \( d_M \) and Euclidean distance \( d_E \) are used to measure the distance regression estimates calculated by \( X \) and \( X^* \). Type I error rate \( R_I \) and Type II error rate \( R_{II} \) are used to evaluate p-values of linear models using encrypted data. More specifically, \( R_I \) is the percent of p-values bigger than 0.05 among the first 100 true significant attributes and \( R_{II} \) is the percent of p-values smaller than 0.05 among the other 400 attributes. The simulations are repeated 100 times.

**TABLE VI: Mean & 95% CI of model estimates caused by noise matrix.**

| \( \sigma \) | \( \rho \) | \( d_M \) | \( d_E \) | \( R_I \) | \( R_{II} \) |
|---|---|---|---|---|---|
| 0.1 | 1.000 | [0.020,0.020] | [0.550,0.562] | 0.004 | 0.050 |
| 0.2 | 0.999 | 0.045 | 1.314 | 0.010 | 0.050 |
| 0.5 | 0.994 | 0.141 | 4.945 | 0.028 | 0.049 |
| 1 | [0.974,0.975] | 0.277 | 11.52 | 0.058 | 0.049 |
| 2 | 0.909 | 0.383 | 18.08 | 0.156 | 0.051 |
| 5 | [0.652,0.661] | 0.418 | 21.60 | 0.564 | 0.050 |

Figure 6 depicts model estimate deviations measured by five metrics and Table VI presents mean and 95% confidence intervals (CIs). As the variance of noise \( \sigma \) increases, decreased Pearson’s correlation coefficient, increased Manhattan distance and Euclidean distance present reduced accuracy of model estimates. Type I error rate increases as the variance of noise \( \sigma \) increases and Type II error rate is not affected.

**VIII. CONCLUSION**

In this paper, we propose efficient privacy preserving schemes for collaborative linear models when data is distributed among different agencies. The proposed schemes are against malicious adversaries while satisfying local differential privacy. Both vertical and horizontal partitioning are investigated. Cross validation is feasible in the proposed schemes. The security proof and the experimental analysis demonstrate that the proposed schemes achieve desirable security and efficiency. In the future, we are interested in extending the privacy preserving schemes to other statistical models.

**APPENDIX A**

**HORIZONTAL PARTITIONING**

Algorithm 3 presents detailed procedures of pre-modeling phase for horizontal partitioning scenario while Figure 7 illustrates procedures with specific encryption orders.

**Algorithm 3: Pre-modeling phase: horizontal partitioning**

**Input:** \( p \times p \) invertible matrix \( B_0 \) and \( 3 \times 3 \) invertible matrix \( C_0 \)

**Output:** Masked dataset \( [X_1^T, \ldots, X_K^T]^T \) and \( [Z_1^T, \ldots, Z_K^T]^T \)

\[
\text{for } \text{Agency } i = 1, 2, \ldots, K \text{ do}
\]

1. generate a random positive integer \( s_i \) and random coefficient vector \( \{b_{i1}, \ldots, b_{is_i}\} \), noise matrix \( \Delta_i \) following \( N(0, \sigma^2) \) and orthogonal matrices \( A_i, A_{i+1}, \ldots, A_{iK} \) with dimension \( n_i \times n_1, n_2 \times n_2, \ldots, n_K \times n_K \), respectively;

2. \( B_i = \sum_{j=1}^{s_i} b_{ij} B_0^j, C_i = \sum_{j=1}^{s_i} b_{ij} C_0^j \);

\[
\text{for } \text{Agency } i = 1, 2, \ldots, K \text{ do}
\]

3. generate \( Q_i \), a permutation of \( \{1, \ldots, K\} \) with \( Q_i(1) = i \);

4. compute \( X_i^* = A_{i1}(X_i + \Delta_i)B_i, Z_i^* = A_{i1}Y_iC_i \) and send to \( Q_i(2) \);

5. for Agency \( Q_i(j) \) (\( j = 2, \ldots, K \)) do

6. compute \( X_i^* = A_{Q_i(j)}(X_i^* + \Delta_i)B_{Q_i(j)} \) and \( Z_i^* = A_{Q_i(j)}(X_i^* + \Delta_i)B_{Q_i(j)} \);

7. if \( j \neq K \) then

8. send \( X_i^* \) and \( Z_i^* \) to agency \( Q_i(j+1) \);

9. else

10. return \( X_i^* \) and \( Z_i^* \);

**APPENDIX B**

**POST-MODELING PHASE**

Decryption procedures for model estimates are included in Algorithm 4.
Algorithm 4: Post-modeling phase

1. if it is vertical partitioning then
2.  denote $\hat{\beta}_i^* = (\hat{\beta}_i^T, \hat{\beta}_i^T, \ldots, \hat{\beta}_K^T)^T$ where $\hat{\beta}_i^*$ are
3.  masked estimates of attributes in agency $i$ for 3 responses;
4.  for Agency $i = 1, 2, \ldots, K$ do
5.      for Agency $Q_i(j)$ $(j = K, \ldots, 1)$ do
6.         if $Q_i(j)$ holds the true response then
7.          compute $\hat{\beta}_i^* = B_{Q_i(j), i}\hat{\beta}_i^* C_i^{-1}$;
8.         else
9.          compute $\hat{\beta}_i^* = B_{Q_i(j), i}\hat{\beta}_i^*$;
10.         if $j > 1$, send $\hat{\beta}_i^*$ to agency $Q_i(j + 1)$;
11.         if $j = 1$, $\beta_i = \hat{\beta}_i^*$;
12.  else if it is horizontal partitioning then
13.  denote $\hat{\beta}^* = [\hat{\beta}_1^*, \ldots, \hat{\beta}_K^*]$ be the $p \times 3K$
14.  encrypted model estimates;
15.  for Agency $i = 1, 2, \ldots, K$ do
16.      generate $Q_i$, a permutation of $\{1, \ldots, K\}$ with
17.      $Q_i(1) = i$;
18.      compute $\hat{\beta}_i^* = B_{Q_i, i}\hat{\beta}_i^* C_i^{-1}$ and send to $Q_i(2)$;
19.      for Agency $Q_i(j)$ $(j = 2, \ldots, K)$ do
20.         compute $\hat{\beta}_i^* = B_{Q_i(j), i}\hat{\beta}_i^*$;
21.         if $j < K$, send $\hat{\beta}_i^*$ to agency $Q_i(j + 1)$;
22.         if $j = K$, $\beta_i = \hat{\beta}_i^*$;
23.  return $\hat{\beta} = \sum_{i=1}^K \hat{\beta}_i$.

APPENDIX C

KNOWN PLAINTEXT ATTACK

Agency $i$ can not recover masking matrices $A$ and $B$ from final encrypted data $X_i^*$ and $Z_i^*$. To make notation simple, let $X_i^* = X^* = AXB$ and $Z_i^* = Z^* = AZ$. We have $X^*B^{-1} = AX$ and $A^{-1}X^* = XB$.

REFERENCES

[1] 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, ser. KDD ’02, New York, NY, USA, 2002, p. 639–644.

[2] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, “Privacy-preserving ridge regression on hundreds of millions of records,” in 2013 IEEE Symposium on Security and Privacy, pp. 334–348, 2013.

[3] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahir, and D. Evans, “Privacy-preserving distributed linear regression on high-dimensional data,” Proceedings on Privacy Enhancing Technologies, vol. 4, pp. 345–364, 2017.

[4] I. Giaconelli, S. Jha, M. Joye, C. Page, and K. Yoon, “Privacy-preserving ridge regression with only linearly-homomorphic encryption,” IACR Cryptology ePrint Archive, pp. 243–261, 06 2018.

[5] P. Mohassel and Y. Zhang, “SecureML: A system for scalable privacy-preserving machine learning,” in 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 19–38.

[6] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, “Helen: Maliciously secure cooperative learning for linear models,” in 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 724–738.
